

libstdc++

Generated by Doxygen 1.8.15

1 Todo List	2
2 Module Documentation	3
2.1 Adaptors for pointers to functions	3
2.1.1 Detailed Description	4
2.1.2 Function Documentation	4
2.2 Adaptors for pointers to members	5
2.2.1 Detailed Description	5
2.3 Algorithms	6
2.3.1 Detailed Description	6
2.4 Allocators	7
2.4.1 Detailed Description	8
2.4.2 Typedef Documentation	8
2.4.3 Function Documentation	8
2.5 Arithmetic Classes	9
2.5.1 Detailed Description	9
2.6 Array creation functions	10
2.6.1 Detailed Description	10
2.6.2 Function Documentation	10
2.7 Associative	11
2.7.1 Detailed Description	11
2.8 Atomics	12
2.8.1 Detailed Description	16
2.8.2 Macro Definition Documentation	16
2.8.3 Typedef Documentation	17
2.8.4 Enumeration Type Documentation	26
2.8.5 Function Documentation	26
2.9 Base and Implementation Classes	27
2.9.1 Detailed Description	29
2.9.2 Function Documentation	29
2.10 Base and Implementation Classes	30
2.10.1 Detailed Description	31
2.10.2 Enumeration Type Documentation	31
2.11 Base and Policy Classes	32
2.11.1 Detailed Description	32
2.12 Base and Policy Classes	33
2.12.1 Detailed Description	33
2.13 Base and Policy Classes	34
2.13.1 Detailed Description	34

2.14 Bernoulli Distributions	35
2.14.1 Detailed Description	36
2.14.2 Function Documentation	36
2.15 Binary Search	40
2.15.1 Detailed Description	40
2.15.2 Function Documentation	41
2.16 Binder Classes	46
2.16.1 Detailed Description	47
2.16.2 Function Documentation	47
2.17 Bit manipulation	49
2.18 Boolean Operations Classes	50
2.18.1 Detailed Description	50
2.19 Branch-Based	51
2.19.1 Detailed Description	51
2.20 Comparison Classes	52
2.20.1 Detailed Description	52
2.21 Complex Numbers	53
2.21.1 Detailed Description	56
2.21.2 Function Documentation	56
2.22 Concurrency	70
2.22.1 Detailed Description	70
2.23 Condition Variables	71
2.23.1 Detailed Description	71
2.23.2 Enumeration Type Documentation	71
2.24 Const-propagating wrapper	72
2.24.1 Detailed Description	73
2.25 Containers	74
2.25.1 Detailed Description	74
2.26 Containers	75
2.26.1 Detailed Description	75
2.27 Data Structure Type	76
2.27.1 Detailed Description	76
2.28 Decimal Floating-Point Arithmetic	77
2.28.1 Detailed Description	77
2.29 Diagnostics	78
2.29.1 Detailed Description	80
2.29.2 Function Documentation	80
2.30 Dynamic Bitset.	82
2.30.1 Detailed Description	83

2.30.2 Function Documentation	83
2.31 Exceptions	87
2.31.1 Detailed Description	88
2.31.2 Typedef Documentation	88
2.31.3 Function Documentation	89
2.32 Exceptions	92
2.32.1 Detailed Description	92
2.33 Extensions	93
2.33.1 Detailed Description	93
2.34 Filesystem TS	94
2.34.1 Detailed Description	98
2.34.2 Enumeration Type Documentation	99
2.35 Function Objects	100
2.35.1 Detailed Description	101
2.35.2 Function Documentation	101
2.36 Futures	102
2.36.1 Detailed Description	103
2.36.2 Enumeration Type Documentation	103
2.36.3 Function Documentation	104
2.37 Generalized Numeric operations	106
2.37.1 Detailed Description	106
2.37.2 Function Documentation	106
2.38 Hash-Based	113
2.38.1 Detailed Description	113
2.39 Hashes	114
2.39.1 Detailed Description	114
2.40 Heap	115
2.40.1 Detailed Description	115
2.40.2 Function Documentation	115
2.41 Heap-Based	122
2.41.1 Detailed Description	123
2.41.2 Function Documentation	123
2.42 I/O	125
2.42.1 Detailed Description	126
2.42.2 Typedef Documentation	126
2.43 Invalidation Guarantees	132
2.43.1 Detailed Description	132
2.44 Iterator Tags	133
2.44.1 Detailed Description	133

2.45 Iterators	134
2.45.1 Detailed Description	137
2.45.2 Function Documentation	137
2.46 Library Fundamentals TS	141
2.46.1 Detailed Description	142
2.47 List-Based	143
2.47.1 Detailed Description	143
2.48 Locales	144
2.48.1 Detailed Description	145
2.48.2 Function Documentation	145
2.49 Mathematical Special Functions	147
2.49.1 Detailed Description	149
2.49.2 Mathematical Special Functions	149
2.49.3 Function Documentation	151
2.50 Memory	184
2.50.1 Detailed Description	184
2.50.2 Function Documentation	185
2.51 Metaprogramming	188
2.51.1 Detailed Description	192
2.51.2 Typedef Documentation	192
2.51.3 Variable Documentation	197
2.52 Mutating	198
2.52.1 Detailed Description	200
2.52.2 Function Documentation	200
2.53 Mutexes	222
2.53.1 Detailed Description	223
2.53.2 Function Documentation	223
2.53.3 Variable Documentation	224
2.54 Negators	226
2.54.1 Detailed Description	226
2.54.2 Function Documentation	226
2.55 Networking-ts	228
2.55.1 Detailed Description	228
2.56 Non-Mutating	229
2.56.1 Detailed Description	230
2.56.2 Function Documentation	230
2.57 Normal Distributions	250
2.57.1 Detailed Description	251
2.57.2 Function Documentation	251

2.58 Numeric Arrays	255
2.58.1 Detailed Description	264
2.58.2 Function Documentation	264
2.59 Numerics	302
2.59.1 Detailed Description	302
2.60 Optional values	303
2.60.1 Detailed Description	303
2.60.2 Variable Documentation	303
2.61 Pointer Abstractions	304
2.61.1 Detailed Description	307
2.61.2 Macro Definition Documentation	307
2.61.3 Function Documentation	307
2.62 Pointer Safety and Garbage Collection	323
2.62.1 Detailed Description	323
2.62.2 Enumeration Type Documentation	323
2.62.3 Function Documentation	324
2.63 Poisson Distributions	325
2.63.1 Detailed Description	326
2.63.2 Function Documentation	326
2.64 Policy-Based Data Structures	332
2.64.1 Detailed Description	332
2.65 Random Number Distributions	333
2.65.1 Detailed Description	333
2.66 Random Number Generation	334
2.66.1 Detailed Description	334
2.66.2 Function Documentation	334
2.67 Random Number Generators	335
2.67.1 Detailed Description	336
2.67.2 Typedef Documentation	336
2.67.3 Function Documentation	337
2.68 Random Number Utilities	341
2.68.1 Detailed Description	341
2.69 Rational Arithmetic	342
2.69.1 Detailed Description	343
2.69.2 Typedef Documentation	343
2.70 Regular Expressions	344
2.70.1 Detailed Description	349
2.70.2 Typedef Documentation	349
2.70.3 Function Documentation	351

2.71 SGI	391
2.71.1 Detailed Description	392
2.71.2 Function Documentation	393
2.72 Sequences	403
2.72.1 Detailed Description	403
2.73 Set Operations	404
2.73.1 Detailed Description	404
2.73.2 Function Documentation	405
2.74 Sorting	413
2.74.1 Detailed Description	415
2.74.2 Function Documentation	415
2.75 Strings	438
2.75.1 Detailed Description	438
2.75.2 Typedef Documentation	438
2.76 TR1 Mathematical Special Functions	440
2.76.1 Detailed Description	442
2.76.2 Function Documentation	442
2.77 Tags	448
2.77.1 Detailed Description	448
2.77.2 Typedef Documentation	448
2.78 Technical Specifications	449
2.78.1 Detailed Description	449
2.79 Threads	450
2.79.1 Detailed Description	450
2.80 Time	451
2.80.1 Detailed Description	453
2.80.2 Typedef Documentation	453
2.80.3 Function Documentation	454
2.81 Traits	457
2.81.1 Detailed Description	458
2.82 Type-safe container of any type	459
2.82.1 Detailed Description	459
2.82.2 Function Documentation	460
2.83 Uniform Distributions	464
2.83.1 Detailed Description	464
2.83.2 Function Documentation	464
2.84 Unordered Associative	468
2.84.1 Detailed Description	468
2.85 Utilities	469

2.85.1 Detailed Description	472
2.85.2 Function Documentation	472
2.85.3 Variable Documentation	483
3 Namespace Documentation	484
3.1 <code>__gnu_cxx</code> Namespace Reference	484
3.1.1 Detailed Description	499
3.1.2 Typedef Documentation	499
3.1.3 Function Documentation	499
3.2 <code>__gnu_cxx::__detail</code> Namespace Reference	512
3.2.1 Detailed Description	513
3.2.2 Function Documentation	513
3.3 <code>__gnu_cxx::typelist</code> Namespace Reference	514
3.3.1 Detailed Description	514
3.3.2 Function Documentation	515
3.4 <code>__gnu_debug</code> Namespace Reference	515
3.4.1 Detailed Description	521
3.4.2 Enumeration Type Documentation	521
3.4.3 Function Documentation	521
3.5 <code>__gnu_internal</code> Namespace Reference	525
3.5.1 Detailed Description	525
3.6 <code>__gnu_parallel</code> Namespace Reference	525
3.6.1 Detailed Description	533
3.6.2 Typedef Documentation	533
3.6.3 Enumeration Type Documentation	534
3.6.4 Function Documentation	535
3.6.5 Variable Documentation	584
3.7 <code>__gnu_pbds</code> Namespace Reference	584
3.7.1 Detailed Description	586
3.8 <code>__gnu_sequential</code> Namespace Reference	586
3.8.1 Detailed Description	586
3.9 <code>abi</code> Namespace Reference	586
3.9.1 Detailed Description	586
3.10 <code>std</code> Namespace Reference	587
3.10.1 Detailed Description	686
3.10.2 Typedef Documentation	687
3.10.3 Enumeration Type Documentation	690
3.10.4 Function Documentation	691
3.10.5 Variable Documentation	793

3.11 std::__debug Namespace Reference	796
3.11.1 Detailed Description	800
3.11.2 Function Documentation	800
3.12 std::__detail Namespace Reference	801
3.12.1 Detailed Description	805
3.12.2 Function Documentation	805
3.13 std::__parallel Namespace Reference	807
3.13.1 Detailed Description	824
3.14 std::chrono Namespace Reference	824
3.14.1 Detailed Description	824
3.15 std::decimal Namespace Reference	825
3.15.1 Detailed Description	833
3.15.2 Function Documentation	834
3.16 std::experimental Namespace Reference	834
3.16.1 Detailed Description	834
3.17 std::literals::chrono_literals Namespace Reference	834
3.17.1 Detailed Description	835
3.17.2 Function Documentation	835
3.18 std::placeholders Namespace Reference	838
3.18.1 Detailed Description	838
3.19 std::regex_constants Namespace Reference	838
3.19.1 Detailed Description	840
3.19.2 Enumeration Type Documentation	840
3.19.3 Function Documentation	842
3.19.4 Variable Documentation	848
3.20 std::rel_ops Namespace Reference	854
3.20.1 Detailed Description	854
3.20.2 Function Documentation	854
3.21 std::this_thread Namespace Reference	856
3.21.1 Detailed Description	856
3.21.2 Function Documentation	856
3.22 std::tr1 Namespace Reference	857
3.22.1 Detailed Description	860
3.23 std::tr1::__detail Namespace Reference	860
3.23.1 Detailed Description	860
3.24 std::tr2 Namespace Reference	860
3.24.1 Detailed Description	862
3.25 std::tr2::__detail Namespace Reference	862
3.25.1 Detailed Description	862

4 Class Documentation	862
4.1 <code>__gnu_parallel::__accumulate_binop_reduct<_BinOp></code> Struct Template Reference	862
4.1.1 Detailed Description	862
4.2 <code>__gnu_parallel::__accumulate_selector<_It></code> Struct Template Reference	863
4.2.1 Detailed Description	863
4.2.2 Member Function Documentation	863
4.2.3 Member Data Documentation	864
4.3 <code>std::__add_pointer_helper<_Tp, bool></code> Struct Template Reference	864
4.3.1 Detailed Description	864
4.4 <code>__gnu_parallel::__adjacent_difference_selector<_It></code> Struct Template Reference	865
4.4.1 Detailed Description	865
4.4.2 Member Data Documentation	865
4.5 <code>__gnu_parallel::__adjacent_find_selector</code> Struct Reference	866
4.5.1 Detailed Description	866
4.5.2 Member Function Documentation	867
4.6 <code>__gnu_cxx::__alloc_traits<_Alloc, typename></code> Struct Template Reference	868
4.6.1 Detailed Description	870
4.6.2 Member Typedef Documentation	870
4.6.3 Member Function Documentation	871
4.7 <code>std::__allocated_ptr<_Alloc></code> Struct Template Reference	878
4.7.1 Detailed Description	878
4.7.2 Constructor & Destructor Documentation	878
4.7.3 Member Function Documentation	879
4.8 <code>std::__atomic_base<_ITp></code> Struct Template Reference	880
4.8.1 Detailed Description	881
4.9 <code>std::__atomic_base<_PTp*></code> Struct Template Reference	882
4.9.1 Detailed Description	882
4.10 <code>std::__atomic_flag_base</code> Struct Reference	883
4.10.1 Detailed Description	883
4.11 <code>std::__basic_future<_Res></code> Class Template Reference	884
4.11.1 Detailed Description	885
4.11.2 Member Typedef Documentation	885
4.11.3 Member Function Documentation	885
4.12 <code>__gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference	886
4.12.1 Detailed Description	887
4.12.2 Member Typedef Documentation	887
4.13 <code>__gnu_parallel::__binder2nd<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType></code> Class Template Reference	888
4.13.1 Detailed Description	888

4.13.2 Member Typedef Documentation	889
4.14 <code>std::__codecv_t_abstract_base< _InternT, _ExternT, _StateT ></code> Class Template Reference	889
4.14.1 Detailed Description	891
4.14.2 Member Function Documentation	891
4.15 <code>__gnu_cxx::__common_pool_policy< _PoolTp, _Thread ></code> Struct Template Reference	894
4.15.1 Detailed Description	895
4.16 <code>__gnu_parallel::__count_if_selector< _It, _Diff ></code> Struct Template Reference	895
4.16.1 Detailed Description	895
4.16.2 Member Function Documentation	896
4.16.3 Member Data Documentation	896
4.17 <code>__gnu_parallel::__count_selector< _It, _Diff ></code> Struct Template Reference	897
4.17.1 Detailed Description	897
4.17.2 Member Function Documentation	897
4.17.3 Member Data Documentation	898
4.18 <code>std::__ctype_abstract_base< _CharT ></code> Class Template Reference	899
4.18.1 Detailed Description	900
4.18.2 Member Typedef Documentation	901
4.18.3 Member Function Documentation	901
4.19 <code>std::__detector< _Default, _AlwaysVoid, _Op, _Args ></code> Struct Template Reference	916
4.19.1 Detailed Description	916
4.20 <code>std::__detector< _Default, __void_t< _Op< _Args... >, _Op, _Args... ></code> Struct Template Reference	917
4.20.1 Detailed Description	917
4.21 <code>std::tr2::__dynamic_bitset_base< _WordT, _Alloc ></code> Struct Template Reference	917
4.21.1 Detailed Description	919
4.21.2 Member Data Documentation	919
4.22 <code>__gnu_parallel::__fill_selector< _It ></code> Struct Template Reference	919
4.22.1 Detailed Description	920
4.22.2 Member Function Documentation	920
4.22.3 Member Data Documentation	920
4.23 <code>__gnu_parallel::__find_first_of_selector< _FIterator ></code> Struct Template Reference	921
4.23.1 Detailed Description	922
4.23.2 Member Function Documentation	922
4.24 <code>__gnu_parallel::__find_if_selector</code> Struct Reference	923
4.24.1 Detailed Description	923
4.24.2 Member Function Documentation	924
4.25 <code>__gnu_parallel::__for_each_selector< _It ></code> Struct Template Reference	925
4.25.1 Detailed Description	925
4.25.2 Member Function Documentation	925
4.25.3 Member Data Documentation	926

4.26	__cxxabiv1::__forced_unwind Class Reference	926
4.26.1	Detailed Description	926
4.27	std::__future_base Struct Reference	927
4.27.1	Detailed Description	928
4.27.2	Member Typedef Documentation	928
4.28	__gnu_parallel::__generate_selector< _It > Struct Template Reference	928
4.28.1	Detailed Description	929
4.28.2	Member Function Documentation	929
4.28.3	Member Data Documentation	929
4.29	__gnu_parallel::__generic_find_selector Struct Reference	930
4.29.1	Detailed Description	930
4.30	__gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference	931
4.30.1	Detailed Description	932
4.30.2	Member Data Documentation	932
4.31	__gnu_parallel::__identity_selector< _It > Struct Template Reference	932
4.31.1	Detailed Description	933
4.31.2	Member Function Documentation	933
4.31.3	Member Data Documentation	934
4.32	__gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference	934
4.32.1	Detailed Description	935
4.32.2	Constructor & Destructor Documentation	935
4.32.3	Member Function Documentation	935
4.32.4	Member Data Documentation	936
4.33	std::__is_location_invariant< _Tp > Struct Template Reference	937
4.33.1	Detailed Description	937
4.34	std::__is_nullptr_t< _Tp > Struct Template Reference	937
4.34.1	Detailed Description	937
4.35	std::__is_tuple_like_impl< std::pair< _T1, _T2 > > Struct Template Reference	938
4.35.1	Detailed Description	938
4.36	__gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference	939
4.36.1	Detailed Description	939
4.37	__gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference	939
4.37.1	Detailed Description	939
4.38	__gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference	940
4.38.1	Detailed Description	940
4.39	__gnu_parallel::__mismatch_selector Struct Reference	941
4.39.1	Detailed Description	941
4.39.2	Member Function Documentation	941
4.40	__gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference	943

4.40.1 Detailed Description	944
4.41 <code>__gnu_cxx::__mt_alloc_base< _Tp ></code> Class Template Reference	944
4.41.1 Detailed Description	945
4.42 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	945
4.42.1 Detailed Description	945
4.43 <code>__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	946
4.43.1 Detailed Description	946
4.44 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	946
4.44.1 Detailed Description	946
4.45 <code>__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	947
4.45.1 Detailed Description	947
4.46 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	947
4.46.1 Detailed Description	947
4.47 <code>__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare ></code> Struct Template Reference	948
4.47.1 Detailed Description	948
4.48 <code>std::__numeric_limits_base</code> Struct Reference	948
4.48.1 Detailed Description	949
4.48.2 Member Data Documentation	949
4.49 <code>__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread ></code> Struct Template Reference	954
4.49.1 Detailed Description	954
4.50 <code>__gnu_cxx::__pool< _Thread ></code> Class Template Reference	955
4.50.1 Detailed Description	955
4.51 <code>__gnu_cxx::__pool< false ></code> Class Template Reference	955
4.51.1 Detailed Description	956
4.52 <code>__gnu_cxx::__pool< true ></code> Class Template Reference	956
4.52.1 Detailed Description	957
4.53 <code>__gnu_cxx::__pool_alloc< _Tp ></code> Class Template Reference	957
4.53.1 Detailed Description	959
4.54 <code>__gnu_cxx::__pool_alloc_base</code> Class Reference	959
4.54.1 Detailed Description	960
4.55 <code>__gnu_cxx::__pool_base</code> Struct Reference	960
4.55.1 Detailed Description	961
4.56 <code>__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc ></code> Class Template Reference	961
4.56.1 Detailed Description	963
4.57 <code>std::tr2::__reflection_typelist< _Elements ></code> Struct Template Reference	964

4.57.1 Detailed Description	964
4.58 <code>std::tr2::__reflection_typelist< _First, _Rest... ></code> Struct Template Reference	964
4.58.1 Detailed Description	964
4.59 <code>std::tr2::__reflection_typelist<></code> Struct Template Reference	964
4.59.1 Detailed Description	965
4.60 <code>__gnu_parallel::__replace_if_selector< _It, _Op, _Tp ></code> Struct Template Reference	965
4.60.1 Detailed Description	966
4.60.2 Constructor & Destructor Documentation	966
4.60.3 Member Function Documentation	966
4.60.4 Member Data Documentation	967
4.61 <code>__gnu_parallel::__replace_selector< _It, _Tp ></code> Struct Template Reference	967
4.61.1 Detailed Description	968
4.61.2 Constructor & Destructor Documentation	968
4.61.3 Member Function Documentation	968
4.61.4 Member Data Documentation	969
4.62 <code>__gnu_cxx::__scoped_lock</code> Class Reference	970
4.62.1 Detailed Description	970
4.63 <code>__gnu_parallel::__transform1_selector< _It ></code> Struct Template Reference	970
4.63.1 Detailed Description	971
4.63.2 Member Function Documentation	971
4.63.3 Member Data Documentation	971
4.64 <code>__gnu_parallel::__transform2_selector< _It ></code> Struct Template Reference	972
4.64.1 Detailed Description	972
4.64.2 Member Function Documentation	973
4.64.3 Member Data Documentation	973
4.65 <code>__gnu_parallel::__unary_negate< _Predicate, argument_type ></code> Class Template Reference	974
4.65.1 Detailed Description	974
4.65.2 Member Typedef Documentation	974
4.66 <code>__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base ></code> Class Template Reference	975
4.66.1 Detailed Description	979
4.66.2 Constructor & Destructor Documentation	979
4.66.3 Member Function Documentation	983
4.66.4 Member Data Documentation	1044
4.67 <code>__gnu_debug::__After_nth_from< _Iterator ></code> Class Template Reference	1045
4.67.1 Detailed Description	1045
4.68 <code>std::_Base_bitset< _Nw ></code> Struct Template Reference	1045
4.68.1 Detailed Description	1046
4.68.2 Member Data Documentation	1047
4.69 <code>std::_Base_bitset< 0 ></code> Struct Template Reference	1047

4.69.1 Detailed Description	1048
4.70 std::_Base_bitset< 1 > Struct Template Reference	1048
4.70.1 Detailed Description	1049
4.71 __gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference	1049
4.71.1 Detailed Description	1049
4.72 std::_Bind< _Signature > Struct Template Reference	1050
4.72.1 Detailed Description	1050
4.73 std::_Bind_result< _Result, _Signature > Struct Template Reference	1050
4.73.1 Detailed Description	1050
4.74 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference	1050
4.74.1 Detailed Description	1051
4.75 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference	1051
4.75.1 Detailed Description	1051
4.76 __gnu_cxx::_Caster< _ToType > Struct Template Reference	1052
4.76.1 Detailed Description	1052
4.77 __gnu_cxx::_Char_types< _CharT > Struct Template Reference	1052
4.77.1 Detailed Description	1052
4.78 __gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference	1053
4.78.1 Detailed Description	1054
4.79 std::__detail::_Compiler< _TraitsT > Class Template Reference	1054
4.79.1 Detailed Description	1055
4.80 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference	1055
4.80.1 Detailed Description	1055
4.81 std::__detail::_Default_ranged_hash Struct Reference	1055
4.81.1 Detailed Description	1055
4.82 std::_Deque_base< _Tp, _Alloc > Class Template Reference	1056
4.82.1 Detailed Description	1057
4.82.2 Member Function Documentation	1057
4.83 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference	1058
4.83.1 Detailed Description	1059
4.83.2 Member Function Documentation	1059
4.84 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference	1060
4.84.1 Detailed Description	1060
4.84.2 Constructor & Destructor Documentation	1061
4.84.3 Member Data Documentation	1061
4.85 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference	1063
4.85.1 Detailed Description	1063
4.85.2 Member Data Documentation	1063

4.86	__gnu_parallel::_DummyReduct Struct Reference	1065
4.86.1	Detailed Description	1065
4.87	std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag > Struct Template Reference	1065
4.87.1	Detailed Description	1065
4.88	std::_Enable_default_constructor< _Switch, _Tag > Struct Template Reference	1066
4.88.1	Detailed Description	1066
4.89	std::_Enable_destructor< _Switch, _Tag > Struct Template Reference	1066
4.89.1	Detailed Description	1066
4.90	std::_Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag > Struct Template Reference	1067
4.90.1	Detailed Description	1067
4.91	__gnu_debug::_Equal_to< _Type > Class Template Reference	1067
4.91.1	Detailed Description	1068
4.92	__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference	1068
4.92.1	Detailed Description	1069
4.92.2	Member Typedef Documentation	1069
4.93	std::_detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1070
4.93.1	Detailed Description	1070
4.94	std::_detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1070
4.94.1	Detailed Description	1071
4.95	std::_detail::_Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1071
4.95.1	Detailed Description	1071
4.96	__gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference	1072
4.96.1	Detailed Description	1072
4.96.2	Member Typedef Documentation	1072
4.97	std::_detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode > Class Template Reference	1073
4.97.1	Detailed Description	1074
4.98	__gnu_cxx::_ExtPtr_allocator< _Tp > Class Template Reference	1074
4.98.1	Detailed Description	1075
4.99	__gnu_cxx::_detail::_Ffit_finder< _Tp > Class Template Reference	1076
4.99.1	Detailed Description	1076
4.99.2	Member Typedef Documentation	1076
4.100	std::_Function_base Class Reference	1077
4.100.1	Detailed Description	1078
4.101	std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference	1078
4.101.1	Detailed Description	1079

4.102	std::_Fwd_list_const_iterator< _Tp > Struct Template Reference	1080
4.102.1	Detailed Description	1080
4.102.2	Friends And Related Function Documentation	1080
4.103	std::_Fwd_list_iterator< _Tp > Struct Template Reference	1081
4.103.1	Detailed Description	1082
4.103.2	Friends And Related Function Documentation	1082
4.104	std::_Fwd_list_node< _Tp > Struct Template Reference	1083
4.104.1	Detailed Description	1083
4.105	std::_Fwd_list_node_base Struct Reference	1084
4.105.1	Detailed Description	1084
4.106	__gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference	1085
4.106.1	Detailed Description	1085
4.106.2	Constructor & Destructor Documentation	1085
4.106.3	Member Function Documentation	1086
4.106.4	Friends And Related Function Documentation	1087
4.107	std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code > Struct Template Reference	1088
4.107.1	Detailed Description	1088
4.108	std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false > Struct Template Reference	1088
4.108.1	Detailed Description	1089
4.109	std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true > Struct Template Reference	1090
4.109.1	Detailed Description	1091
4.110	std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > Struct Template Reference	1091
4.110.1	Detailed Description	1092
4.111	std::__detail::_Hash_node< _Value, _Cache_hash_code > Struct Template Reference	1092
4.111.1	Detailed Description	1092
4.112	std::__detail::_Hash_node< _Value, false > Struct Template Reference	1093
4.112.1	Detailed Description	1094
4.113	std::__detail::_Hash_node< _Value, true > Struct Template Reference	1094
4.113.1	Detailed Description	1095
4.114	std::__detail::_Hash_node_base Struct Reference	1095
4.114.1	Detailed Description	1096
4.115	std::__detail::_Hash_node_value_base< _Value > Struct Template Reference	1096
4.115.1	Detailed Description	1097
4.116	std::_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits > Class Template Reference	1097
4.116.1	Detailed Description	1101

4.117	<code>std::__detail::_Hashtable_alloc<_NodeAlloc></code> Struct Template Reference	1103
4.117.1	Detailed Description	1104
4.118	<code>std::__detail::_Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits></code> Struct Template Reference	1104
4.118.1	Detailed Description	1105
4.119	<code>std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, __use_ebo></code> Struct Template Reference	1106
4.119.1	Detailed Description	1106
4.120	<code>std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false></code> Struct Template Reference	1106
4.120.1	Detailed Description	1106
4.121	<code>std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, true></code> Struct Template Reference	1106
4.121.1	Detailed Description	1107
4.122	<code>std::__detail::_Hashtable_traits<_Cache_hash_code, _Constant_iterators, _Unique_keys></code> Struct Template Reference	1107
4.122.1	Detailed Description	1107
4.123	<code>__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata></code> Struct Template Reference	1108
4.123.1	Detailed Description	1109
4.124	<code>__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata></code> Struct Template Reference	1109
4.124.1	Detailed Description	1110
4.125	<code>std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators></code> Struct Template Reference	1111
4.125.1	Detailed Description	1111
4.126	<code>std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false></code> Struct Template Reference	1111
4.126.1	Detailed Description	1113
4.127	<code>std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true></code> Struct Template Reference	1113
4.127.1	Detailed Description	1114
4.128	<code>std::__detail::_Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits></code> Struct Template Reference	1115
4.128.1	Detailed Description	1116
4.129	<code>__gnu_cxx::_Invalid_type</code> Struct Reference	1116
4.129.1	Detailed Description	1116
4.130	<code>__gnu_pbds::detail::pat_trie_base::_Iter<Node, Leaf, Head, Inode, Is_Forward_Iterator></code> Class Template Reference	1116
4.130.1	Detailed Description	1118
4.131	<code>__gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory></code> Class Template Reference	1118
4.131.1	Detailed Description	1119
4.131.2	Member Typedef Documentation	1120
4.131.3	Member Function Documentation	1120
4.131.4	Friends And Related Function Documentation	1120
4.131.5	Member Data Documentation	1123

4.132 __gnu_parallel::IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference	1123
4.132.1 Detailed Description	1124
4.133 __gnu_parallel::Job< _DifferenceTp > Struct Template Reference	1124
4.133.1 Detailed Description	1125
4.133.2 Member Data Documentation	1125
4.134 __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference	1126
4.134.1 Detailed Description	1127
4.135 __gnu_parallel::Less< _T1, _T2 > Struct Template Reference	1127
4.135.1 Detailed Description	1128
4.135.2 Member Typedef Documentation	1128
4.136 __gnu_parallel::Lexicographic< _T1, _T2, _Compare > Class Template Reference	1129
4.136.1 Detailed Description	1130
4.136.2 Member Typedef Documentation	1130
4.137 __gnu_parallel::LexicographicReverse< _T1, _T2, _Compare > Class Template Reference	1131
4.137.1 Detailed Description	1131
4.137.2 Member Typedef Documentation	1131
4.138 std::_List_base< _Tp, _Alloc > Class Template Reference	1132
4.138.1 Detailed Description	1134
4.139 std::_List_const_iterator< _Tp > Struct Template Reference	1134
4.139.1 Detailed Description	1135
4.140 std::_List_iterator< _Tp > Struct Template Reference	1135
4.140.1 Detailed Description	1136
4.141 std::_List_node< _Tp > Struct Template Reference	1136
4.141.1 Detailed Description	1137
4.142 std::__detail::_List_node_base Struct Reference	1137
4.142.1 Detailed Description	1138
4.143 std::__detail::_List_node_header Struct Reference	1138
4.143.1 Detailed Description	1139
4.144 std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_↵ iterators, __cache > Struct Template Reference	1139
4.144.1 Detailed Description	1140
4.145 std::__detail::_Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_↵ iterators, __cache > Struct Template Reference	1140
4.145.1 Detailed Description	1141
4.146 std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_↵ code > Struct Template Reference	1141
4.146.1 Detailed Description	1141
4.147 std::__detail::_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true > Struct Template Reference	1142

4.147.1 Detailed Description	1143
4.148 <code>__gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser</code> Struct Reference	1143
4.148.1 Detailed Description	1143
4.148.2 Member Data Documentation	1143
4.149 <code>__gnu_parallel::LoserTreePointerBase< _Tp, _Compare >::Loser</code> Struct Reference	1144
4.149.1 Detailed Description	1144
4.150 <code>__gnu_parallel::LoserTree< __stable, _Tp, _Compare ></code> Class Template Reference	1145
4.150.1 Detailed Description	1145
4.150.2 Member Function Documentation	1146
4.150.3 Member Data Documentation	1147
4.151 <code>__gnu_parallel::LoserTree< false, _Tp, _Compare ></code> Class Template Reference	1147
4.151.1 Detailed Description	1148
4.151.2 Member Function Documentation	1148
4.152 <code>__gnu_parallel::LoserTreeBase< _Tp, _Compare ></code> Class Template Reference	1150
4.152.1 Detailed Description	1151
4.152.2 Constructor & Destructor Documentation	1151
4.152.3 Member Function Documentation	1152
4.152.4 Member Data Documentation	1153
4.153 <code>__gnu_parallel::LoserTreePointer< __stable, _Tp, _Compare ></code> Class Template Reference	1154
4.153.1 Detailed Description	1155
4.154 <code>__gnu_parallel::LoserTreePointer< false, _Tp, _Compare ></code> Class Template Reference	1155
4.154.1 Detailed Description	1156
4.155 <code>__gnu_parallel::LoserTreePointerBase< _Tp, _Compare ></code> Class Template Reference	1156
4.155.1 Detailed Description	1157
4.156 <code>__gnu_parallel::LoserTreePointerUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	1157
4.156.1 Detailed Description	1158
4.157 <code>__gnu_parallel::LoserTreePointerUnguarded< false, _Tp, _Compare ></code> Class Template Reference	1158
4.157.1 Detailed Description	1159
4.158 <code>__gnu_parallel::LoserTreePointerUnguardedBase< _Tp, _Compare ></code> Class Template Reference	1159
4.158.1 Detailed Description	1160
4.159 <code>__gnu_parallel::LoserTreeTraits< _Tp ></code> Struct Template Reference	1160
4.159.1 Detailed Description	1160
4.159.2 Member Data Documentation	1161
4.160 <code>__gnu_parallel::LoserTreeUnguarded< __stable, _Tp, _Compare ></code> Class Template Reference	1161
4.160.1 Detailed Description	1162
4.161 <code>__gnu_parallel::LoserTreeUnguarded< false, _Tp, _Compare ></code> Class Template Reference	1162
4.161.1 Detailed Description	1163
4.162 <code>__gnu_parallel::LoserTreeUnguardedBase< _Tp, _Compare ></code> Class Template Reference	1163
4.162.1 Detailed Description	1164

4.163	std::__detail::_Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys > Struct Template Reference	1164
4.163.1	Detailed Description	1165
4.164	std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false > Struct Template Reference	1165
4.164.1	Detailed Description	1165
4.165	std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true > Struct Template Reference	1165
4.165.1	Detailed Description	1166
4.166	std::__detail::_Mask_range_hashing Struct Reference	1166
4.166.1	Detailed Description	1166
4.167	__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference	1167
4.167.1	Detailed Description	1167
4.168	__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference	1167
4.168.1	Detailed Description	1167
4.169	std::__detail::_Mod_range_hashing Struct Reference	1168
4.169.1	Detailed Description	1168
4.170	std::_Mu< _Arg, _IsBindExp, _IsPlaceholder > Class Template Reference	1168
4.170.1	Detailed Description	1168
4.171	std::_Mu< _Arg, false, false > Class Template Reference	1169
4.171.1	Detailed Description	1169
4.172	std::_Mu< _Arg, false, true > Class Template Reference	1169
4.172.1	Detailed Description	1169
4.173	std::_Mu< _Arg, true, false > Class Template Reference	1170
4.173.1	Detailed Description	1170
4.174	std::_Mu< reference_wrapper< _Tp >, false, false > Class Template Reference	1170
4.174.1	Detailed Description	1170
4.175	__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference	1171
4.175.1	Detailed Description	1171
4.175.2	Member Typedef Documentation	1171
4.176	__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference	1172
4.176.1	Detailed Description	1173
4.177	__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference	1173
4.177.1	Detailed Description	1174
4.177.2	Member Typedef Documentation	1175
4.177.3	Member Function Documentation	1175
4.178	std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache > Struct Template Reference	1177
4.178.1	Detailed Description	1178

4.179	__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >	
	Class Template Reference	1178
4.179.1	Detailed Description	1179
4.179.2	Member Typedef Documentation	1180
4.179.3	Member Function Documentation	1180
4.180	std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >	Struct Template Reference 1182
4.180.1	Detailed Description	1183
4.181	std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >	Struct Template Reference 1183
4.181.1	Detailed Description	1184
4.182	__gnu_debug::_Not_equal_to< _Type >	Class Template Reference 1184
4.182.1	Detailed Description	1184
4.183	std::_Not_fn< _Fn >	Class Template Reference 1184
4.183.1	Detailed Description	1185
4.184	__gnu_parallel::_Nothing	Struct Reference 1185
4.184.1	Detailed Description	1185
4.184.2	Member Function Documentation	1185
4.185	__gnu_parallel::_Piece< _DifferenceTp >	Struct Template Reference 1186
4.185.1	Detailed Description	1186
4.185.2	Member Data Documentation	1186
4.186	std::_Placeholder< _Num >	Struct Template Reference 1187
4.186.1	Detailed Description	1187
4.187	__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >	Struct Template Reference 1187
4.187.1	Detailed Description	1188
4.187.2	Member Typedef Documentation	1188
4.188	__gnu_parallel::_PMWMSSortingData< _RAIter >	Struct Template Reference 1189
4.188.1	Detailed Description	1189
4.188.2	Member Data Documentation	1189
4.189	__gnu_cxx::_Pointer_adapter< _Storage_policy >	Class Template Reference 1191
4.189.1	Detailed Description	1193
4.190	std::__detail::_Power2_rehash_policy	Struct Reference 1194
4.190.1	Detailed Description	1195
4.191	std::__detail::_Prime_rehash_policy	Struct Reference 1195
4.191.1	Detailed Description	1195
4.192	__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >	Class Template Reference 1196
4.192.1	Detailed Description	1196
4.192.2	Constructor & Destructor Documentation	1196
4.192.3	Member Function Documentation	1197
4.193	__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >	Class Template Reference 1197
4.193.1	Detailed Description	1197

4.194	__gnu_parallel::__QSBThreadLocal< _RAIter > Struct Template Reference	1198
4.194.1	Detailed Description	1198
4.194.2	Member Typedef Documentation	1198
4.194.3	Constructor & Destructor Documentation	1199
4.194.4	Member Data Documentation	1199
4.195	std::__detail::__Quoted_string< _String, _CharT > Struct Template Reference	1201
4.195.1	Detailed Description	1201
4.196	__gnu_parallel::__RandomNumber Class Reference	1201
4.196.1	Detailed Description	1201
4.196.2	Constructor & Destructor Documentation	1201
4.196.3	Member Function Documentation	1202
4.197	std::__detail::__Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _↵ RehashPolicy, _Traits, typename > Struct Template Reference	1203
4.197.1	Detailed Description	1203
4.198	std::__detail::__Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _↵ RehashPolicy, _Traits, false_type > Struct Template Reference	1204
4.198.1	Detailed Description	1204
4.199	std::__detail::__Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _↵ RehashPolicy, _Traits, true_type > Struct Template Reference	1204
4.199.1	Detailed Description	1204
4.200	__gnu_cxx::__Relative_pointer_impl< _Tp > Class Template Reference	1205
4.200.1	Detailed Description	1205
4.201	__gnu_cxx::__Relative_pointer_impl< const _Tp > Class Template Reference	1205
4.201.1	Detailed Description	1206
4.202	__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference	1206
4.202.1	Detailed Description	1206
4.202.2	Constructor & Destructor Documentation	1206
4.202.3	Member Function Documentation	1207
4.203	std::__future_base::__Result< _Res > Struct Template Reference	1208
4.203.1	Detailed Description	1209
4.204	std::__future_base::__Result< _Res & > Struct Template Reference	1209
4.204.1	Detailed Description	1210
4.205	std::__future_base::__Result< void > Struct Template Reference	1210
4.205.1	Detailed Description	1211
4.206	std::__future_base::__Result_alloc< _Res, _Alloc > Struct Template Reference	1211
4.206.1	Detailed Description	1212
4.207	std::__future_base::__Result_base Struct Reference	1212
4.207.1	Detailed Description	1213
4.208	__gnu_debug::__Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware > Class Template Reference	1213

4.208.1 Detailed Description	1213
4.209 __gnu_debug::_Safe_forward_list<_SafeSequence> Class Template Reference	1214
4.209.1 Detailed Description	1215
4.209.2 Member Function Documentation	1215
4.209.3 Member Data Documentation	1216
4.210 __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category> Class Template Reference	1217
4.210.1 Detailed Description	1219
4.210.2 Constructor & Destructor Documentation	1220
4.210.3 Member Function Documentation	1221
4.210.4 Member Data Documentation	1228
4.211 __gnu_debug::_Safe_iterator_base Class Reference	1230
4.211.1 Detailed Description	1231
4.211.2 Constructor & Destructor Documentation	1231
4.211.3 Member Function Documentation	1232
4.211.4 Member Data Documentation	1234
4.212 __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence> Class Template Reference	1235
4.212.1 Detailed Description	1237
4.212.2 Constructor & Destructor Documentation	1237
4.212.3 Member Function Documentation	1239
4.212.4 Member Data Documentation	1246
4.213 __gnu_debug::_Safe_local_iterator_base Class Reference	1247
4.213.1 Detailed Description	1248
4.213.2 Constructor & Destructor Documentation	1248
4.213.3 Member Function Documentation	1249
4.213.4 Member Data Documentation	1251
4.214 __gnu_debug::_Safe_node_sequence<_Sequence> Class Template Reference	1253
4.214.1 Detailed Description	1254
4.214.2 Member Function Documentation	1254
4.214.3 Member Data Documentation	1256
4.215 __gnu_debug::_Safe_sequence<_Sequence> Class Template Reference	1257
4.215.1 Detailed Description	1258
4.215.2 Member Function Documentation	1258
4.215.3 Member Data Documentation	1260
4.216 __gnu_debug::_Safe_sequence_base Class Reference	1261
4.216.1 Detailed Description	1262
4.216.2 Constructor & Destructor Documentation	1262
4.216.3 Member Function Documentation	1262
4.216.4 Member Data Documentation	1264
4.217 __gnu_debug::_Safe_unordered_container<_Container> Class Template Reference	1265

4.217.1 Detailed Description	1266
4.217.2 Member Function Documentation	1266
4.217.3 Member Data Documentation	1268
4.218 <code>__gnu_debug::_Safe_unordered_container_base</code> Class Reference	1269
4.218.1 Detailed Description	1270
4.218.2 Constructor & Destructor Documentation	1270
4.218.3 Member Function Documentation	1270
4.218.4 Member Data Documentation	1272
4.219 <code>__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence ></code> Class Template Reference	1273
4.219.1 Detailed Description	1273
4.220 <code>__gnu_parallel::SamplingSorter<__stable, _RAIter, _StrictWeakOrdering ></code> Struct Template Reference	1274
4.220.1 Detailed Description	1274
4.221 <code>__gnu_parallel::SamplingSorter<false, _RAIter, _StrictWeakOrdering ></code> Struct Template Reference	1274
4.221.1 Detailed Description	1274
4.222 <code>std::__detail::_Scanner<_CharT ></code> Class Template Reference	1274
4.222.1 Detailed Description	1276
4.222.2 Member Enumeration Documentation	1276
4.223 <code>__gnu_debug::Sequence_traits<_Sequence ></code> Struct Template Reference	1276
4.223.1 Detailed Description	1277
4.224 <code>__gnu_parallel::Settings</code> Struct Reference	1277
4.224.1 Detailed Description	1278
4.224.2 Member Function Documentation	1278
4.224.3 Member Data Documentation	1279
4.225 <code>std::_Sp_ebo_helper<_Nm, _Tp, false ></code> Struct Template Reference	1288
4.225.1 Detailed Description	1288
4.226 <code>std::_Sp_ebo_helper<_Nm, _Tp, true ></code> Struct Template Reference	1288
4.226.1 Detailed Description	1288
4.227 <code>__gnu_parallel::SplitConsistently<__exact, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1289
4.227.1 Detailed Description	1289
4.228 <code>__gnu_parallel::SplitConsistently<false, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1289
4.228.1 Detailed Description	1289
4.229 <code>__gnu_parallel::SplitConsistently<true, _RAIter, _Compare, _SortingPlacesIterator ></code> Struct Template Reference	1289
4.229.1 Detailed Description	1290
4.230 <code>std::__detail::_StateSeq<_TraitsT ></code> Class Template Reference	1290
4.230.1 Detailed Description	1290
4.231 <code>__gnu_cxx::Std_pointer_impl<_Tp ></code> Class Template Reference	1291
4.231.1 Detailed Description	1291

4.232 <code>std::_Temporary_buffer<_ForwardIterator, _Tp></code> Class Template Reference	1291
4.232.1 Detailed Description	1292
4.232.2 Constructor & Destructor Documentation	1292
4.232.3 Member Function Documentation	1293
4.233 <code>std::_Tuple_impl<_Idx, _Elements></code> Struct Template Reference	1294
4.233.1 Detailed Description	1294
4.234 <code>std::_Tuple_impl<_Idx, _Head, _Tail...></code> Struct Template Reference	1294
4.234.1 Detailed Description	1296
4.235 <code>__gnu_cxx::_Unqualified_type<_Tp></code> Struct Template Reference	1296
4.235.1 Detailed Description	1296
4.236 <code>std::_Vector_base<_Tp, _Alloc></code> Struct Template Reference	1296
4.236.1 Detailed Description	1297
4.237 <code>std::add_const<_Tp></code> Struct Template Reference	1297
4.237.1 Detailed Description	1298
4.238 <code>std::add_cv<_Tp></code> Struct Template Reference	1298
4.238.1 Detailed Description	1298
4.239 <code>std::add_lvalue_reference<_Tp></code> Struct Template Reference	1298
4.239.1 Detailed Description	1299
4.240 <code>std::add_rvalue_reference<_Tp></code> Struct Template Reference	1299
4.240.1 Detailed Description	1299
4.241 <code>std::add_volatile<_Tp></code> Struct Template Reference	1299
4.241.1 Detailed Description	1300
4.242 <code>std::adopt_lock_t</code> Struct Reference	1300
4.242.1 Detailed Description	1300
4.243 <code>std::aligned_storage<_Len, _Align></code> Struct Template Reference	1300
4.243.1 Detailed Description	1300
4.244 <code>std::aligned_union<_Len, _Types></code> Struct Template Reference	1301
4.244.1 Detailed Description	1301
4.244.2 Member Typedef Documentation	1301
4.245 <code>std::alignment_of<_Tp></code> Struct Template Reference	1302
4.245.1 Detailed Description	1302
4.246 <code>std::allocator<_Tp></code> Class Template Reference	1303
4.246.1 Detailed Description	1304
4.247 <code>std::allocator<void></code> Class Template Reference	1304
4.247.1 Detailed Description	1304
4.248 <code>std::allocator_arg_t</code> Struct Reference	1305
4.248.1 Detailed Description	1305
4.249 <code>std::allocator_traits<_Alloc></code> Struct Template Reference	1305
4.249.1 Detailed Description	1306

4.249.2 Member Typedef Documentation	1306
4.249.3 Member Function Documentation	1309
4.250 std::allocator_traits< allocator< _Tp > > Struct Template Reference	1313
4.250.1 Detailed Description	1314
4.250.2 Member Typedef Documentation	1314
4.250.3 Member Function Documentation	1316
4.251 __gnu_cxx::limit_condition::always_adjustor Struct Reference	1320
4.251.1 Detailed Description	1320
4.252 __gnu_cxx::random_condition::always_adjustor Struct Reference	1320
4.252.1 Detailed Description	1320
4.253 __gnu_cxx::annotate_base Struct Reference	1321
4.253.1 Detailed Description	1321
4.254 std::experimental::fundamentals_v1::any Class Reference	1322
4.254.1 Detailed Description	1322
4.254.2 Constructor & Destructor Documentation	1322
4.254.3 Member Function Documentation	1324
4.255 std::array< _Tp, _Nm > Struct Template Reference	1326
4.255.1 Detailed Description	1327
4.256 __gnu_pbds::associative_tag Struct Reference	1328
4.256.1 Detailed Description	1328
4.257 std::atomic< _Tp > Struct Template Reference	1328
4.257.1 Detailed Description	1329
4.258 std::atomic< _Tp * > Struct Template Reference	1329
4.258.1 Detailed Description	1331
4.259 std::atomic< bool > Struct Template Reference	1331
4.259.1 Detailed Description	1332
4.260 std::atomic< char > Struct Template Reference	1332
4.260.1 Detailed Description	1334
4.261 std::atomic< char16_t > Struct Template Reference	1334
4.261.1 Detailed Description	1336
4.262 std::atomic< char32_t > Struct Template Reference	1336
4.262.1 Detailed Description	1338
4.263 std::atomic< int > Struct Template Reference	1338
4.263.1 Detailed Description	1340
4.264 std::atomic< long > Struct Template Reference	1340
4.264.1 Detailed Description	1342
4.265 std::atomic< long long > Struct Template Reference	1342
4.265.1 Detailed Description	1344
4.266 std::atomic< short > Struct Template Reference	1344

4.266.1 Detailed Description	1346
4.267 std::atomic< signed char > Struct Template Reference	1346
4.267.1 Detailed Description	1348
4.268 std::atomic< unsigned char > Struct Template Reference	1348
4.268.1 Detailed Description	1350
4.269 std::atomic< unsigned int > Struct Template Reference	1350
4.269.1 Detailed Description	1352
4.270 std::atomic< unsigned long > Struct Template Reference	1352
4.270.1 Detailed Description	1354
4.271 std::atomic< unsigned long long > Struct Template Reference	1354
4.271.1 Detailed Description	1356
4.272 std::atomic< unsigned short > Struct Template Reference	1356
4.272.1 Detailed Description	1358
4.273 std::atomic< wchar_t > Struct Template Reference	1358
4.273.1 Detailed Description	1360
4.274 std::atomic_flag Struct Reference	1360
4.274.1 Detailed Description	1361
4.275 std::auto_ptr< _Tp > Class Template Reference	1361
4.275.1 Detailed Description	1362
4.275.2 Member Typedef Documentation	1362
4.275.3 Constructor & Destructor Documentation	1362
4.275.4 Member Function Documentation	1364
4.276 std::auto_ptr_ref< _Tp1 > Struct Template Reference	1367
4.276.1 Detailed Description	1367
4.277 std::back_insert_iterator< _Container > Class Template Reference	1368
4.277.1 Detailed Description	1369
4.277.2 Member Typedef Documentation	1369
4.277.3 Constructor & Destructor Documentation	1370
4.277.4 Member Function Documentation	1370
4.278 std::bad_alloc Class Reference	1372
4.278.1 Detailed Description	1372
4.278.2 Member Function Documentation	1372
4.279 std::experimental::fundamentals_v1::bad_any_cast Class Reference	1373
4.279.1 Detailed Description	1373
4.279.2 Member Function Documentation	1374
4.280 std::bad_cast Class Reference	1374
4.280.1 Detailed Description	1375
4.280.2 Member Function Documentation	1375
4.281 std::bad_exception Class Reference	1375

4.281.1 Detailed Description	1376
4.281.2 Member Function Documentation	1376
4.282 std::bad_function_call Class Reference	1376
4.282.1 Detailed Description	1377
4.282.2 Member Function Documentation	1377
4.283 std::experimental::fundamentals_v1::bad_optional_access Class Reference	1377
4.283.1 Detailed Description	1378
4.283.2 Member Function Documentation	1378
4.284 std::bad_typeid Class Reference	1378
4.284.1 Detailed Description	1379
4.284.2 Member Function Documentation	1379
4.285 std::bad_weak_ptr Class Reference	1379
4.285.1 Detailed Description	1380
4.285.2 Member Function Documentation	1380
4.286 __gnu_parallel::balanced_quicksort_tag Struct Reference	1380
4.286.1 Detailed Description	1381
4.286.2 Member Function Documentation	1381
4.287 __gnu_parallel::balanced_tag Struct Reference	1382
4.287.1 Detailed Description	1382
4.287.2 Member Function Documentation	1382
4.288 std::tr2::bases< _Tp > Struct Template Reference	1383
4.288.1 Detailed Description	1383
4.289 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_TI, _Alloc > Class Template Reference	1383
4.289.1 Detailed Description	1384
4.290 __gnu_pbds::basic_branch_tag Struct Reference	1385
4.290.1 Detailed Description	1385
4.291 std::basic_filebuf< _CharT, _Traits > Class Template Reference	1385
4.291.1 Detailed Description	1388
4.291.2 Constructor & Destructor Documentation	1389
4.291.3 Member Function Documentation	1389
4.291.4 Member Data Documentation	1408
4.292 std::basic_fstream< _CharT, _Traits > Class Template Reference	1413
4.292.1 Detailed Description	1419
4.292.2 Member Typedef Documentation	1420
4.292.3 Member Enumeration Documentation	1422
4.292.4 Constructor & Destructor Documentation	1423
4.292.5 Member Function Documentation	1424
4.292.6 Member Data Documentation	1472

4.293 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference	1480
4.293.1 Detailed Description	1480
4.294 __gnu_pbds::basic_hash_tag Struct Reference	1481
4.294.1 Detailed Description	1482
4.295 std::basic_ifstream< _CharT, _Traits > Class Template Reference	1482
4.295.1 Detailed Description	1487
4.295.2 Member Typedef Documentation	1488
4.295.3 Member Enumeration Documentation	1490
4.295.4 Constructor & Destructor Documentation	1491
4.295.5 Member Function Documentation	1492
4.295.6 Member Data Documentation	1528
4.296 __gnu_pbds::basic_invalidation_guarantee Struct Reference	1536
4.296.1 Detailed Description	1536
4.297 std::basic_ios< _CharT, _Traits > Class Template Reference	1537
4.297.1 Detailed Description	1540
4.297.2 Member Typedef Documentation	1540
4.297.3 Member Enumeration Documentation	1545
4.297.4 Constructor & Destructor Documentation	1545
4.297.5 Member Function Documentation	1546
4.297.6 Member Data Documentation	1561
4.298 std::basic_iostream< _CharT, _Traits > Class Template Reference	1569
4.298.1 Detailed Description	1575
4.298.2 Member Typedef Documentation	1576
4.298.3 Member Enumeration Documentation	1578
4.298.4 Constructor & Destructor Documentation	1579
4.298.5 Member Function Documentation	1579
4.298.6 Member Data Documentation	1626
4.299 std::basic_istream< _CharT, _Traits > Class Template Reference	1634
4.299.1 Detailed Description	1639
4.299.2 Member Typedef Documentation	1640
4.299.3 Member Enumeration Documentation	1642
4.299.4 Constructor & Destructor Documentation	1643
4.299.5 Member Function Documentation	1643
4.299.6 Member Data Documentation	1678
4.300 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference	1686
4.300.1 Detailed Description	1691
4.300.2 Member Typedef Documentation	1692
4.300.3 Member Enumeration Documentation	1694

4.300.4 Constructor & Destructor Documentation	1695
4.300.5 Member Function Documentation	1696
4.300.6 Member Data Documentation	1732
4.301 std::basic_ofstream< _CharT, _Traits > Class Template Reference	1740
4.301.1 Detailed Description	1745
4.301.2 Member Typedef Documentation	1745
4.301.3 Member Enumeration Documentation	1748
4.301.4 Constructor & Destructor Documentation	1748
4.301.5 Member Function Documentation	1749
4.301.6 Member Data Documentation	1777
4.302 std::basic_ostream< _CharT, _Traits > Class Template Reference	1785
4.302.1 Detailed Description	1790
4.302.2 Member Typedef Documentation	1790
4.302.3 Member Enumeration Documentation	1793
4.302.4 Constructor & Destructor Documentation	1794
4.302.5 Member Function Documentation	1794
4.302.6 Member Data Documentation	1821
4.303 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference	1829
4.303.1 Detailed Description	1833
4.303.2 Member Typedef Documentation	1834
4.303.3 Member Enumeration Documentation	1836
4.303.4 Constructor & Destructor Documentation	1837
4.303.5 Member Function Documentation	1838
4.303.6 Member Data Documentation	1865
4.304 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference	1873
4.304.1 Detailed Description	1874
4.304.2 Constructor & Destructor Documentation	1875
4.304.3 Member Function Documentation	1879
4.305 std::basic_streambuf< _CharT, _Traits > Class Template Reference	1887
4.305.1 Detailed Description	1889
4.305.2 Member Typedef Documentation	1890
4.305.3 Constructor & Destructor Documentation	1892
4.305.4 Member Function Documentation	1893
4.305.5 Member Data Documentation	1908
4.306 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference	1910
4.306.1 Detailed Description	1914
4.306.2 Constructor & Destructor Documentation	1915
4.306.3 Member Function Documentation	1919
4.306.4 Member Data Documentation	1975

4.307	__gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference	1975
4.307.1	Detailed Description	1980
4.307.2	Member Function Documentation	1980
4.307.3	Member Data Documentation	2005
4.308	std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits > Class Template Reference	2007
4.308.1	Detailed Description	2008
4.309	std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference	2009
4.309.1	Detailed Description	2011
4.309.2	Constructor & Destructor Documentation	2012
4.309.3	Member Function Documentation	2013
4.309.4	Member Data Documentation	2030
4.310	std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference	2032
4.310.1	Detailed Description	2038
4.310.2	Member Typedef Documentation	2039
4.310.3	Member Enumeration Documentation	2042
4.310.4	Constructor & Destructor Documentation	2042
4.310.5	Member Function Documentation	2043
4.310.6	Member Data Documentation	2090
4.311	std::bernoulli_distribution Class Reference	2098
4.311.1	Detailed Description	2099
4.311.2	Member Typedef Documentation	2099
4.311.3	Constructor & Destructor Documentation	2099
4.311.4	Member Function Documentation	2100
4.311.5	Friends And Related Function Documentation	2101
4.312	std::bidirectional_iterator_tag Struct Reference	2102
4.312.1	Detailed Description	2102
4.313	__gnu_pbds::detail::bin_search_tree_const_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference	2103
4.313.1	Detailed Description	2104
4.314	__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	2104
4.314.1	Detailed Description	2105
4.314.2	Member Typedef Documentation	2105
4.314.3	Member Function Documentation	2107
4.315	__gnu_pbds::detail::bin_search_tree_it< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference	2109
4.315.1	Detailed Description	2110
4.316	__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference	2111
4.316.1	Detailed Description	2111

4.316.2 Member Typedef Documentation	2112
4.316.3 Member Function Documentation	2113
4.317 <code>__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc ></code> Struct Template Reference	2115
4.317.1 Detailed Description	2115
4.317.2 Member Typedef Documentation	2116
4.318 <code>__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc ></code> Struct Template Reference	2116
4.318.1 Detailed Description	2116
4.318.2 Member Typedef Documentation	2117
4.319 <code>__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 ></code> Class Template Reference	2117
4.319.1 Detailed Description	2118
4.319.2 Member Typedef Documentation	2118
4.320 <code>std::binary_function< _Arg1, _Arg2, _Result ></code> Struct Template Reference	2119
4.320.1 Detailed Description	2119
4.320.2 Member Typedef Documentation	2119
4.321 <code>__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc ></code> Class Template Reference	2120
4.321.1 Detailed Description	2122
4.322 <code>__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc ></code> Class Template Reference	2122
4.322.1 Detailed Description	2123
4.322.2 Member Typedef Documentation	2123
4.322.3 Constructor & Destructor Documentation	2125
4.322.4 Member Function Documentation	2126
4.323 <code>__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc ></code> Class Template Reference	2128
4.323.1 Detailed Description	2129
4.323.2 Member Typedef Documentation	2129
4.323.3 Constructor & Destructor Documentation	2130
4.323.4 Member Function Documentation	2131
4.324 <code>__gnu_pbds::binary_heap_tag</code> Struct Reference	2132
4.324.1 Detailed Description	2133
4.325 <code>std::binary_negate< _Predicate ></code> Class Template Reference	2133
4.325.1 Detailed Description	2134
4.325.2 Member Typedef Documentation	2134
4.326 <code>std::binder1st< _Operation ></code> Class Template Reference	2135
4.326.1 Detailed Description	2135
4.326.2 Member Typedef Documentation	2136
4.327 <code>std::binder2nd< _Operation ></code> Class Template Reference	2136
4.327.1 Detailed Description	2137

4.327.2 Member Typedef Documentation	2137
4.328 std::binomial_distribution< _IntType > Class Template Reference	2138
4.328.1 Detailed Description	2139
4.328.2 Member Typedef Documentation	2139
4.328.3 Member Function Documentation	2139
4.328.4 Friends And Related Function Documentation	2142
4.329 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . .	2143
4.329.1 Detailed Description	2145
4.330 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	2145
4.330.1 Detailed Description	2147
4.331 __gnu_pbds::binomial_heap_tag Struct Reference	2148
4.331.1 Detailed Description	2148
4.332 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference	2148
4.332.1 Detailed Description	2149
4.332.2 Member Function Documentation	2150
4.333 std::bitset< _Nb > Class Template Reference	2151
4.333.1 Detailed Description	2154
4.333.2 Constructor & Destructor Documentation	2155
4.333.3 Member Function Documentation	2157
4.334 std::__debug::bitset< _Nb > Class Template Reference	2166
4.334.1 Detailed Description	2167
4.335 std::tr2::bool_set Class Reference	2167
4.335.1 Detailed Description	2168
4.335.2 Constructor & Destructor Documentation	2168
4.335.3 Member Function Documentation	2169
4.336 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc > Struct Template Reference	2170
4.336.1 Detailed Description	2171
4.337 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc > Struct Template Reference . . .	2171
4.337.1 Detailed Description	2172
4.338 std::cauchy_distribution< _RealType > Class Template Reference	2172
4.338.1 Detailed Description	2173
4.338.2 Member Typedef Documentation	2173
4.338.3 Member Function Documentation	2173
4.338.4 Friends And Related Function Documentation	2175
4.339 __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference	2175
4.339.1 Detailed Description	2176
4.339.2 Member Enumeration Documentation	2176
4.339.3 Constructor & Destructor Documentation	2177

4.339.4 Member Function Documentation	2177
4.340 <code>__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc ></code> Class Template Reference	2182
4.340.1 Detailed Description	2183
4.340.2 Constructor & Destructor Documentation	2183
4.341 <code>__gnu_pbds::cc_hash_tag</code> Struct Reference	2187
4.341.1 Detailed Description	2188
4.342 <code>__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy ></code> Class Template Reference	2188
4.342.1 Detailed Description	2190
4.342.2 Member Enumeration Documentation	2190
4.342.3 Member Function Documentation	2191
4.343 <code>__gnu_cxx::char_traits< _CharT ></code> Struct Template Reference	2193
4.343.1 Detailed Description	2194
4.344 <code>std::char_traits< _CharT ></code> Struct Template Reference	2195
4.344.1 Detailed Description	2196
4.345 <code>std::char_traits< __gnu_cxx::character< _Value, _Int, _St > ></code> Struct Template Reference	2196
4.345.1 Detailed Description	2197
4.346 <code>std::char_traits< char ></code> Struct Template Reference	2197
4.346.1 Detailed Description	2197
4.347 <code>std::char_traits< wchar_t ></code> Struct Template Reference	2198
4.347.1 Detailed Description	2198
4.348 <code>__gnu_cxx::character< _Value, _Int, _St ></code> Struct Template Reference	2198
4.348.1 Detailed Description	2199
4.349 <code>std::chi_squared_distribution< _RealType ></code> Class Template Reference	2199
4.349.1 Detailed Description	2200
4.349.2 Member Typedef Documentation	2200
4.349.3 Member Function Documentation	2201
4.349.4 Friends And Related Function Documentation	2203
4.350 <code>std::codecvt< _InternT, _ExternT, _StateT ></code> Class Template Reference	2204
4.350.1 Detailed Description	2206
4.350.2 Member Function Documentation	2206
4.351 <code>std::codecvt< _InternT, _ExternT, encoding_state ></code> Class Template Reference	2210
4.351.1 Detailed Description	2211
4.351.2 Member Function Documentation	2211
4.352 <code>std::codecvt< char, char, mbstate_t ></code> Class Template Reference	2215
4.352.1 Detailed Description	2216
4.352.2 Member Function Documentation	2216
4.353 <code>std::codecvt< char16_t, char, mbstate_t ></code> Class Template Reference	2220
4.353.1 Detailed Description	2221

4.353.2 Member Function Documentation	2221
4.354 std::codecvt< char32_t, char, mbstate_t > Class Template Reference	2225
4.354.1 Detailed Description	2226
4.354.2 Member Function Documentation	2226
4.355 std::codecvt< wchar_t, char, mbstate_t > Class Template Reference	2230
4.355.1 Detailed Description	2231
4.355.2 Member Function Documentation	2231
4.356 std::codecvt_base Class Reference	2235
4.356.1 Detailed Description	2235
4.357 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference	2236
4.357.1 Detailed Description	2237
4.357.2 Member Function Documentation	2237
4.358 std::collate< _CharT > Class Template Reference	2241
4.358.1 Detailed Description	2243
4.358.2 Member Typedef Documentation	2243
4.358.3 Constructor & Destructor Documentation	2243
4.358.4 Member Function Documentation	2244
4.358.5 Member Data Documentation	2248
4.359 std::collate_byname< _CharT > Class Template Reference	2248
4.359.1 Detailed Description	2250
4.359.2 Member Typedef Documentation	2250
4.359.3 Member Function Documentation	2250
4.359.4 Member Data Documentation	2254
4.360 std::common_type< _Tp > Struct Template Reference	2254
4.360.1 Detailed Description	2254
4.361 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference	2255
4.361.1 Detailed Description	2255
4.362 std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference	2255
4.362.1 Detailed Description	2255
4.363 std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference	2255
4.363.1 Detailed Description	2256
4.364 std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference	2256
4.364.1 Detailed Description	2256
4.365 std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference	2256
4.365.1 Detailed Description	2257
4.366 std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > > Struct Template Reference	2257

4.366.1 Detailed Description	2257
4.367 std::complex< _Tp > Struct Template Reference	2257
4.367.1 Detailed Description	2258
4.367.2 Member Typedef Documentation	2258
4.367.3 Constructor & Destructor Documentation	2259
4.367.4 Member Function Documentation	2259
4.368 std::complex< double > Struct Template Reference	2260
4.368.1 Detailed Description	2261
4.369 std::complex< float > Struct Template Reference	2261
4.369.1 Detailed Description	2262
4.370 std::complex< long double > Struct Template Reference	2262
4.370.1 Detailed Description	2263
4.371 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference	2263
4.371.1 Detailed Description	2263
4.372 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference	2264
4.372.1 Detailed Description	2264
4.373 __gnu_cxx::condition_base Struct Reference	2264
4.373.1 Detailed Description	2265
4.374 std::condition_variable Class Reference	2265
4.374.1 Detailed Description	2266
4.375 std::_V2::condition_variable_any Class Reference	2266
4.375.1 Detailed Description	2266
4.376 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference	2266
4.376.1 Detailed Description	2267
4.377 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference	2267
4.377.1 Detailed Description	2268
4.378 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2268
4.378.1 Detailed Description	2269
4.378.2 Member Typedef Documentation	2269
4.379 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2270
4.379.1 Detailed Description	2271
4.379.2 Member Typedef Documentation	2271
4.380 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2272
4.380.1 Detailed Description	2272
4.380.2 Member Typedef Documentation	2272
4.381 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference	2273
4.381.1 Detailed Description	2274
4.381.2 Member Typedef Documentation	2274

4.382	__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference	2274
4.382.1	Detailed Description	2275
4.383	__gnu_parallel::constant_size_blocks_tag Struct Reference	2275
4.383.1	Detailed Description	2276
4.384	__gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference	2276
4.384.1	Detailed Description	2276
4.385	__gnu_cxx::constant_void_fun< _Result > Struct Template Reference	2277
4.385.1	Detailed Description	2277
4.386	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_TI > Struct Template Reference	2277
4.386.1	Detailed Description	2277
4.387	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference	2278
4.387.1	Detailed Description	2278
4.387.2	Member Typedef Documentation	2278
4.388	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference	2278
4.388.1	Detailed Description	2279
4.388.2	Member Typedef Documentation	2279
4.389	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference	2279
4.389.1	Detailed Description	2279
4.389.2	Member Typedef Documentation	2280
4.390	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference	2280
4.390.1	Detailed Description	2280
4.390.2	Member Typedef Documentation	2280
4.391	__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference	2281
4.391.1	Detailed Description	2281
4.391.2	Member Typedef Documentation	2281
4.392	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_TI > Struct Template Reference	2281
4.392.1	Detailed Description	2282
4.392.2	Member Typedef Documentation	2282
4.393	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_TI > Struct Template Reference	2282
4.393.1	Detailed Description	2282
4.393.2	Member Typedef Documentation	2283
4.394	__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_TI > Struct Template Reference	2283
4.394.1	Detailed Description	2283

4.394.2 Member Typedef Documentation	2283
4.395 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_TI > Struct</code> Template Reference	2284
4.395.1 Detailed Description	2284
4.395.2 Member Typedef Documentation	2284
4.396 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_TI > Struct</code> Template Reference	2284
4.396.1 Detailed Description	2285
4.397 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_TI > Struct</code> Template Reference	2285
4.397.1 Detailed Description	2285
4.397.2 Member Typedef Documentation	2285
4.398 <code>__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_TI ></code> Struct Template Reference	2286
4.398.1 Detailed Description	2286
4.398.2 Member Typedef Documentation	2286
4.399 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_TI > Struct</code> Template Reference	2286
4.399.1 Detailed Description	2287
4.399.2 Member Typedef Documentation	2287
4.400 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_TI > Struct</code> Template Reference	2287
4.400.1 Detailed Description	2287
4.400.2 Member Typedef Documentation	2288
4.401 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI ></code> Struct Template Reference	2288
4.401.1 Detailed Description	2288
4.401.2 Member Typedef Documentation	2288
4.402 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_TI > Struct</code> Template Reference	2289
4.402.1 Detailed Description	2289
4.402.2 Member Typedef Documentation	2289
4.403 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI > Struct</code> Template Reference	2289
4.403.1 Detailed Description	2290
4.403.2 Member Typedef Documentation	2290
4.404 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI > Struct</code> Template Reference	2290
4.404.1 Detailed Description	2290
4.405 <code>__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI ></code> Struct Template Reference	2291
4.405.1 Detailed Description	2291

4.405.2 Member Typedef Documentation	2291
4.406 <code>__gnu_pbds::container_error</code> Struct Reference	2292
4.406.1 Detailed Description	2292
4.406.2 Member Function Documentation	2292
4.407 <code>__gnu_pbds::container_tag</code> Struct Reference	2293
4.407.1 Detailed Description	2293
4.408 <code>__gnu_pbds::container_traits< Cntnr ></code> Struct Template Reference	2293
4.408.1 Detailed Description	2294
4.408.2 Member Enumeration Documentation	2294
4.409 <code>__gnu_pbds::container_traits_base< _Tag ></code> Struct Template Reference	2294
4.409.1 Detailed Description	2294
4.410 <code>__gnu_pbds::container_traits_base< binary_heap_tag ></code> Struct Template Reference	2295
4.410.1 Detailed Description	2295
4.411 <code>__gnu_pbds::container_traits_base< binomial_heap_tag ></code> Struct Template Reference	2295
4.411.1 Detailed Description	2296
4.412 <code>__gnu_pbds::container_traits_base< cc_hash_tag ></code> Struct Template Reference	2296
4.412.1 Detailed Description	2296
4.413 <code>__gnu_pbds::container_traits_base< gp_hash_tag ></code> Struct Template Reference	2296
4.413.1 Detailed Description	2297
4.414 <code>__gnu_pbds::container_traits_base< list_update_tag ></code> Struct Template Reference	2297
4.414.1 Detailed Description	2297
4.415 <code>__gnu_pbds::container_traits_base< ov_tree_tag ></code> Struct Template Reference	2297
4.415.1 Detailed Description	2298
4.416 <code>__gnu_pbds::container_traits_base< pairing_heap_tag ></code> Struct Template Reference	2298
4.416.1 Detailed Description	2298
4.417 <code>__gnu_pbds::container_traits_base< pat_trie_tag ></code> Struct Template Reference	2298
4.417.1 Detailed Description	2299
4.418 <code>__gnu_pbds::container_traits_base< rb_tree_tag ></code> Struct Template Reference	2299
4.418.1 Detailed Description	2299
4.419 <code>__gnu_pbds::container_traits_base< rc_binomial_heap_tag ></code> Struct Template Reference	2299
4.419.1 Detailed Description	2300
4.420 <code>__gnu_pbds::container_traits_base< splay_tree_tag ></code> Struct Template Reference	2300
4.420.1 Detailed Description	2300
4.421 <code>__gnu_pbds::container_traits_base< thin_heap_tag ></code> Struct Template Reference	2300
4.421.1 Detailed Description	2301
4.422 <code>std::ctype< _CharT ></code> Class Template Reference	2301
4.422.1 Detailed Description	2303
4.422.2 Member Function Documentation	2303
4.422.3 Member Data Documentation	2318

4.423 std::ctype< char > Class Template Reference	2319
4.423.1 Detailed Description	2321
4.423.2 Member Typedef Documentation	2321
4.423.3 Constructor & Destructor Documentation	2322
4.423.4 Member Function Documentation	2323
4.423.5 Member Data Documentation	2335
4.424 std::ctype< wchar_t > Class Template Reference	2336
4.424.1 Detailed Description	2338
4.424.2 Member Typedef Documentation	2338
4.424.3 Constructor & Destructor Documentation	2338
4.424.4 Member Function Documentation	2339
4.424.5 Member Data Documentation	2354
4.425 std::ctype_base Struct Reference	2354
4.425.1 Detailed Description	2355
4.426 std::ctype_byname< _CharT > Class Template Reference	2355
4.426.1 Detailed Description	2357
4.426.2 Member Function Documentation	2357
4.426.3 Member Data Documentation	2372
4.427 std::ctype_byname< char > Class Template Reference	2373
4.427.1 Detailed Description	2375
4.427.2 Member Typedef Documentation	2375
4.427.3 Member Function Documentation	2375
4.427.4 Member Data Documentation	2388
4.428 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference	2389
4.428.1 Detailed Description	2389
4.429 std::decay< _Tp > Class Template Reference	2390
4.429.1 Detailed Description	2390
4.430 std::decimal::decimal128 Class Reference	2390
4.430.1 Detailed Description	2391
4.430.2 Constructor & Destructor Documentation	2391
4.431 std::decimal::decimal32 Class Reference	2392
4.431.1 Detailed Description	2393
4.431.2 Constructor & Destructor Documentation	2393
4.432 std::decimal::decimal64 Class Reference	2394
4.432.1 Detailed Description	2395
4.432.2 Constructor & Destructor Documentation	2395
4.433 __gnu_pbds::detail::default_comb_hash_fn Struct Reference	2395
4.433.1 Detailed Description	2396
4.433.2 Member Typedef Documentation	2396

4.434	std::default_delete< _Tp > Struct Template Reference	2396
4.434.1	Detailed Description	2396
4.434.2	Constructor & Destructor Documentation	2396
4.434.3	Member Function Documentation	2397
4.435	std::default_delete< _Tp[]> Struct Template Reference	2397
4.435.1	Detailed Description	2398
4.435.2	Constructor & Destructor Documentation	2398
4.435.3	Member Function Documentation	2398
4.436	__gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference	2399
4.436.1	Detailed Description	2399
4.436.2	Member Typedef Documentation	2399
4.437	__gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference	2400
4.437.1	Detailed Description	2400
4.437.2	Member Typedef Documentation	2400
4.438	__gnu_parallel::default_parallel_tag Struct Reference	2400
4.438.1	Detailed Description	2401
4.438.2	Member Function Documentation	2401
4.439	__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference	2402
4.439.1	Detailed Description	2402
4.439.2	Member Typedef Documentation	2402
4.440	__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference	2402
4.440.1	Detailed Description	2403
4.440.2	Member Typedef Documentation	2403
4.441	__gnu_pbds::detail::default_trie_access_traits< Key > Struct Template Reference	2403
4.441.1	Detailed Description	2403
4.442	__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference	2404
4.442.1	Detailed Description	2404
4.442.2	Member Typedef Documentation	2404
4.443	__gnu_pbds::detail::default_update_policy Struct Reference	2404
4.443.1	Detailed Description	2404
4.443.2	Member Typedef Documentation	2405
4.444	std::defer_lock_t Struct Reference	2405
4.444.1	Detailed Description	2405
4.445	std::deque< _Tp, _Alloc > Class Template Reference	2405
4.445.1	Detailed Description	2409
4.445.2	Constructor & Destructor Documentation	2411
4.445.3	Member Function Documentation	2414
4.446	std::__debug::deque< _Tp, _Allocator > Class Template Reference	2436

4.446.1 Detailed Description	2438
4.446.2 Member Function Documentation	2439
4.446.3 Member Data Documentation	2440
4.447 std::tr2::direct_bases< _Tp > Struct Template Reference	2441
4.447.1 Detailed Description	2441
4.448 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference	2442
4.448.1 Detailed Description	2442
4.448.2 Member Function Documentation	2443
4.449 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference	2443
4.449.1 Detailed Description	2444
4.449.2 Member Function Documentation	2444
4.450 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference	2444
4.450.1 Detailed Description	2445
4.450.2 Member Typedef Documentation	2446
4.450.3 Constructor & Destructor Documentation	2446
4.450.4 Member Function Documentation	2448
4.450.5 Friends And Related Function Documentation	2450
4.451 std::discrete_distribution< _IntType > Class Template Reference	2451
4.451.1 Detailed Description	2452
4.451.2 Member Typedef Documentation	2453
4.451.3 Member Function Documentation	2453
4.451.4 Friends And Related Function Documentation	2455
4.452 std::divides< _Tp > Struct Template Reference	2456
4.452.1 Detailed Description	2457
4.452.2 Member Typedef Documentation	2457
4.453 std::divides< void > Struct Template Reference	2458
4.453.1 Detailed Description	2458
4.454 std::domain_error Class Reference	2458
4.454.1 Detailed Description	2459
4.454.2 Member Function Documentation	2459
4.455 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference	2459
4.455.1 Detailed Description	2460
4.456 std::chrono::duration< _Rep, _Period > Struct Template Reference	2460
4.456.1 Detailed Description	2461
4.457 std::chrono::duration_values< _Rep > Struct Template Reference	2461
4.457.1 Detailed Description	2462
4.458 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference	2462
4.458.1 Detailed Description	2466
4.458.2 Constructor & Destructor Documentation	2466

4.458.3 Member Function Documentation	2469
4.459 std::enable_if< bool, _Tp > Struct Template Reference	2482
4.459.1 Detailed Description	2482
4.460 std::enable_shared_from_this< _Tp > Class Template Reference	2482
4.460.1 Detailed Description	2483
4.461 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference	2483
4.461.1 Detailed Description	2486
4.461.2 Member Function Documentation	2486
4.461.3 Member Data Documentation	2505
4.462 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference	2509
4.462.1 Detailed Description	2510
4.463 __gnu_cxx::encoding_state Class Reference	2510
4.463.1 Detailed Description	2511
4.464 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference	2511
4.464.1 Detailed Description	2511
4.465 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference	2511
4.465.1 Detailed Description	2512
4.466 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true > Struct Template Reference	2512
4.466.1 Detailed Description	2512
4.466.2 Member Typedef Documentation	2512
4.467 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference	2513
4.467.1 Detailed Description	2513
4.468 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false > Struct Template Reference	2513
4.468.1 Detailed Description	2513
4.469 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true > Struct Template Reference	2513
4.469.1 Detailed Description	2514
4.470 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference	2514
4.470.1 Detailed Description	2514
4.471 __gnu_parallel::equal_split_tag Struct Reference	2514
4.471.1 Detailed Description	2515
4.472 std::equal_to< _Tp > Struct Template Reference	2515
4.472.1 Detailed Description	2516
4.472.2 Member Typedef Documentation	2516
4.473 std::equal_to< void > Struct Template Reference	2517
4.473.1 Detailed Description	2517
4.474 std::_V2::error_category Class Reference	2517
4.474.1 Detailed Description	2518
4.475 std::error_code Struct Reference	2518
4.475.1 Detailed Description	2519

4.476	std::error_condition Struct Reference	2519
4.476.1	Detailed Description	2520
4.477	__gnu_parallel::exact_tag Struct Reference	2520
4.477.1	Detailed Description	2520
4.477.2	Member Function Documentation	2521
4.478	std::exception Class Reference	2522
4.478.1	Detailed Description	2522
4.479	std::__exception_ptr::exception_ptr Class Reference	2523
4.479.1	Detailed Description	2523
4.479.2	Friends And Related Function Documentation	2524
4.480	std::exponential_distribution<_RealType> Class Template Reference	2524
4.480.1	Detailed Description	2525
4.480.2	Member Typedef Documentation	2525
4.480.3	Constructor & Destructor Documentation	2525
4.480.4	Member Function Documentation	2526
4.480.5	Friends And Related Function Documentation	2528
4.481	std::extent<typename, _Uint> Struct Template Reference	2528
4.481.1	Detailed Description	2529
4.482	std::extreme_value_distribution<_RealType> Class Template Reference	2529
4.482.1	Detailed Description	2530
4.482.2	Member Typedef Documentation	2530
4.482.3	Member Function Documentation	2531
4.482.4	Friends And Related Function Documentation	2533
4.483	std::locale::facet Class Reference	2534
4.483.1	Detailed Description	2535
4.483.2	Constructor & Destructor Documentation	2535
4.484	std::ios_base::failure Class Reference	2536
4.484.1	Detailed Description	2536
4.484.2	Member Function Documentation	2537
4.485	std::experimental::filesystem::v1::filesystem_error Class Reference	2537
4.485.1	Detailed Description	2538
4.485.2	Member Function Documentation	2538
4.486	__gnu_parallel::find_tag Struct Reference	2538
4.486.1	Detailed Description	2539
4.487	std::fisher_f_distribution<_RealType> Class Template Reference	2539
4.487.1	Detailed Description	2540
4.487.2	Member Typedef Documentation	2540
4.487.3	Member Function Documentation	2540
4.487.4	Friends And Related Function Documentation	2542

4.488	__gnu_cxx::forced_error Struct Reference	2544
4.488.1	Detailed Description	2544
4.489	std::forward_iterator_tag Struct Reference	2545
4.489.1	Detailed Description	2545
4.490	std::__debug::forward_list< _Tp, _Alloc > Class Template Reference	2545
4.490.1	Detailed Description	2548
4.490.2	Member Function Documentation	2548
4.490.3	Member Data Documentation	2549
4.491	std::forward_list< _Tp, _Alloc > Class Template Reference	2550
4.491.1	Detailed Description	2553
4.491.2	Constructor & Destructor Documentation	2553
4.491.3	Member Function Documentation	2557
4.492	std::fpos< _StateT > Class Template Reference	2576
4.492.1	Detailed Description	2576
4.492.2	Constructor & Destructor Documentation	2577
4.492.3	Member Function Documentation	2577
4.493	__gnu_cxx::free_list Class Reference	2579
4.493.1	Detailed Description	2580
4.493.2	Member Function Documentation	2580
4.494	std::from_chars_result Struct Reference	2581
4.494.1	Detailed Description	2581
4.495	std::front_insert_iterator< _Container > Class Template Reference	2581
4.495.1	Detailed Description	2582
4.495.2	Member Typedef Documentation	2582
4.495.3	Constructor & Destructor Documentation	2584
4.495.4	Member Function Documentation	2584
4.496	std::function< _Res(_ArgTypes...) > Class Template Reference	2585
4.496.1	Detailed Description	2587
4.496.2	Constructor & Destructor Documentation	2587
4.496.3	Member Function Documentation	2589
4.497	std::future< _Res > Class Template Reference	2594
4.497.1	Detailed Description	2596
4.497.2	Member Typedef Documentation	2596
4.497.3	Constructor & Destructor Documentation	2596
4.497.4	Member Function Documentation	2596
4.498	std::future< _Res & > Class Template Reference	2597
4.498.1	Detailed Description	2598
4.498.2	Member Typedef Documentation	2598
4.498.3	Constructor & Destructor Documentation	2599

4.498.4 Member Function Documentation	2599
4.499 std::future< void > Class Template Reference	2600
4.499.1 Detailed Description	2601
4.499.2 Member Typedef Documentation	2601
4.499.3 Constructor & Destructor Documentation	2602
4.499.4 Member Function Documentation	2602
4.500 std::future_error Class Reference	2603
4.500.1 Detailed Description	2603
4.500.2 Member Function Documentation	2603
4.501 std::gamma_distribution< _RealType > Class Template Reference	2604
4.501.1 Detailed Description	2605
4.501.2 Member Typedef Documentation	2605
4.501.3 Constructor & Destructor Documentation	2605
4.501.4 Member Function Documentation	2606
4.501.5 Friends And Related Function Documentation	2608
4.502 std::geometric_distribution< _IntType > Class Template Reference	2609
4.502.1 Detailed Description	2610
4.502.2 Member Typedef Documentation	2610
4.502.3 Member Function Documentation	2611
4.502.4 Friends And Related Function Documentation	2613
4.503 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference	2613
4.503.1 Detailed Description	2614
4.503.2 Constructor & Destructor Documentation	2615
4.504 __gnu_pbds::gp_hash_tag Struct Reference	2620
4.504.1 Detailed Description	2620
4.505 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference	2621
4.505.1 Detailed Description	2623
4.505.2 Member Enumeration Documentation	2623
4.505.3 Member Function Documentation	2623
4.506 std::greater< _Tp > Struct Template Reference	2626
4.506.1 Detailed Description	2627
4.506.2 Member Typedef Documentation	2627
4.507 std::greater< void > Struct Template Reference	2628
4.507.1 Detailed Description	2628
4.508 std::greater_equal< _Tp > Struct Template Reference	2629
4.508.1 Detailed Description	2629
4.508.2 Member Typedef Documentation	2629
4.509 std::greater_equal< void > Struct Template Reference	2630

4.509.1 Detailed Description	2631
4.510 <code>__gnu_cxx::random_condition::group_adjustor</code> Struct Reference	2631
4.510.1 Detailed Description	2631
4.511 <code>__gnu_parallel::growing_blocks_tag</code> Struct Reference	2631
4.511.1 Detailed Description	2632
4.512 <code>std::gslice</code> Class Reference	2632
4.512.1 Detailed Description	2632
4.513 <code>std::gslice_array< _Tp ></code> Class Template Reference	2633
4.513.1 Detailed Description	2634
4.514 <code>std::has_virtual_destructor< _Tp ></code> Struct Template Reference	2634
4.514.1 Detailed Description	2635
4.515 <code>std::hash< _Tp ></code> Struct Template Reference	2635
4.515.1 Detailed Description	2635
4.516 <code>std::hash< __debug::bitset< _Nb > ></code> Struct Template Reference	2635
4.516.1 Detailed Description	2636
4.517 <code>std::hash< __debug::vector< bool, _Alloc > ></code> Struct Template Reference	2636
4.517.1 Detailed Description	2636
4.518 <code>std::hash< __gnu_cxx::__u16vstring ></code> Struct Template Reference	2637
4.518.1 Detailed Description	2637
4.519 <code>std::hash< __gnu_cxx::__u32vstring ></code> Struct Template Reference	2637
4.519.1 Detailed Description	2638
4.520 <code>std::hash< __gnu_cxx::__vstring ></code> Struct Template Reference	2638
4.520.1 Detailed Description	2638
4.521 <code>std::hash< __gnu_cxx::__wvstring ></code> Struct Template Reference	2638
4.521.1 Detailed Description	2639
4.522 <code>std::hash< __gnu_cxx::throw_value_limit ></code> Struct Template Reference	2639
4.522.1 Detailed Description	2640
4.522.2 Member Typedef Documentation	2640
4.523 <code>std::hash< __gnu_cxx::throw_value_random ></code> Struct Template Reference	2641
4.523.1 Detailed Description	2641
4.523.2 Member Typedef Documentation	2641
4.524 <code>std::hash< __shared_ptr< _Tp, _Lp > ></code> Struct Template Reference	2642
4.524.1 Detailed Description	2642
4.525 <code>std::hash< _Tp * ></code> Struct Template Reference	2643
4.525.1 Detailed Description	2643
4.526 <code>std::hash< bool ></code> Struct Template Reference	2643
4.526.1 Detailed Description	2644
4.527 <code>std::hash< char ></code> Struct Template Reference	2644
4.527.1 Detailed Description	2644

4.528 std::hash< char16_t > Struct Template Reference	2644
4.528.1 Detailed Description	2645
4.529 std::hash< char32_t > Struct Template Reference	2645
4.529.1 Detailed Description	2645
4.530 std::hash< double > Struct Template Reference	2646
4.530.1 Detailed Description	2646
4.531 std::hash< error_code > Struct Template Reference	2646
4.531.1 Detailed Description	2647
4.532 std::hash< experimental::optional< _Tp > > Struct Template Reference	2647
4.532.1 Detailed Description	2647
4.533 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference	2647
4.533.1 Detailed Description	2648
4.534 std::hash< float > Struct Template Reference	2648
4.534.1 Detailed Description	2648
4.535 std::hash< int > Struct Template Reference	2649
4.535.1 Detailed Description	2649
4.536 std::hash< long > Struct Template Reference	2649
4.536.1 Detailed Description	2650
4.537 std::hash< long double > Struct Template Reference	2650
4.537.1 Detailed Description	2650
4.538 std::hash< long long > Struct Template Reference	2650
4.538.1 Detailed Description	2651
4.539 std::hash< shared_ptr< _Tp > > Struct Template Reference	2651
4.539.1 Detailed Description	2651
4.540 std::hash< short > Struct Template Reference	2652
4.540.1 Detailed Description	2652
4.541 std::hash< signed char > Struct Template Reference	2652
4.541.1 Detailed Description	2653
4.542 std::hash< string > Struct Template Reference	2653
4.542.1 Detailed Description	2653
4.543 std::hash< thread::id > Struct Template Reference	2653
4.543.1 Detailed Description	2654
4.544 std::hash< type_index > Struct Template Reference	2654
4.544.1 Detailed Description	2654
4.545 std::hash< u16string > Struct Template Reference	2655
4.545.1 Detailed Description	2655
4.546 std::hash< u32string > Struct Template Reference	2655
4.546.1 Detailed Description	2656
4.547 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference	2656

4.547.1 Detailed Description	2656
4.548 std::hash< unsigned char > Struct Template Reference	2656
4.548.1 Detailed Description	2657
4.549 std::hash< unsigned int > Struct Template Reference	2657
4.549.1 Detailed Description	2657
4.550 std::hash< unsigned long > Struct Template Reference	2658
4.550.1 Detailed Description	2658
4.551 std::hash< unsigned long long > Struct Template Reference	2658
4.551.1 Detailed Description	2659
4.552 std::hash< unsigned short > Struct Template Reference	2659
4.552.1 Detailed Description	2659
4.553 std::hash< wchar_t > Struct Template Reference	2659
4.553.1 Detailed Description	2660
4.554 std::hash< wstring > Struct Template Reference	2660
4.554.1 Detailed Description	2660
4.555 std::hash<::bitset< _Nb > > Struct Template Reference	2661
4.555.1 Detailed Description	2661
4.556 std::hash<::vector< bool, _Alloc > > Struct Template Reference	2661
4.556.1 Detailed Description	2662
4.557 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference	2662
4.557.1 Detailed Description	2662
4.558 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference	2663
4.558.1 Detailed Description	2663
4.559 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference	2663
4.559.1 Detailed Description	2664
4.560 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference	2664
4.560.1 Detailed Description	2664
4.560.2 Constructor & Destructor Documentation	2664
4.561 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference	2665
4.561.1 Detailed Description	2666
4.561.2 Member Enumeration Documentation	2666
4.561.3 Constructor & Destructor Documentation	2667
4.561.4 Member Function Documentation	2667
4.562 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference	2668
4.562.1 Detailed Description	2668
4.563 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference	2669
4.563.1 Detailed Description	2669

4.564	__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2669
4.564.1	Detailed Description	2671
4.565	__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference	2671
4.565.1	Detailed Description	2673
4.566	__gnu_cxx::hash_multiset< _Value, _HashFn, _EqualKey, _Alloc > Class Template Reference	2673
4.566.1	Detailed Description	2674
4.567	__gnu_pbds::hash_prime_size_policy Class Reference	2675
4.567.1	Detailed Description	2675
4.567.2	Member Typedef Documentation	2675
4.567.3	Constructor & Destructor Documentation	2675
4.568	__gnu_cxx::hash_set< _Value, _HashFn, _EqualKey, _Alloc > Class Template Reference	2676
4.568.1	Detailed Description	2677
4.569	__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference	2678
4.569.1	Detailed Description	2679
4.569.2	Constructor & Destructor Documentation	2679
4.569.3	Member Function Documentation	2680
4.570	std::thread::id Class Reference	2682
4.570.1	Detailed Description	2682
4.571	std::locale::id Class Reference	2682
4.571.1	Detailed Description	2683
4.571.2	Constructor & Destructor Documentation	2683
4.571.3	Friends And Related Function Documentation	2683
4.572	std::experimental::fundamentals_v1::in_place_t Struct Reference	2684
4.572.1	Detailed Description	2684
4.573	std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference	2685
4.573.1	Detailed Description	2685
4.573.2	Member Typedef Documentation	2686
4.573.3	Constructor & Destructor Documentation	2686
4.573.4	Member Function Documentation	2688
4.573.5	Friends And Related Function Documentation	2690
4.574	std::indirect_array< _Tp > Class Template Reference	2691
4.574.1	Detailed Description	2692
4.575	std::initializer_list< _E > Class Template Reference	2693
4.575.1	Detailed Description	2693
4.575.2	Friends And Related Function Documentation	2693
4.576	std::input_iterator_tag Struct Reference	2695
4.576.1	Detailed Description	2695
4.577	__gnu_pbds::insert_error Struct Reference	2696

4.577.1 Detailed Description	2696
4.577.2 Member Function Documentation	2696
4.578 std::insert_iterator< _Container > Class Template Reference	2697
4.578.1 Detailed Description	2698
4.578.2 Member Typedef Documentation	2698
4.578.3 Constructor & Destructor Documentation	2699
4.578.4 Member Function Documentation	2700
4.579 std::integer_sequence< _Tp, _Idx > Struct Template Reference	2701
4.579.1 Detailed Description	2701
4.580 std::integral_constant< _Tp, __v > Struct Template Reference	2702
4.580.1 Detailed Description	2703
4.581 std::invalid_argument Class Reference	2703
4.581.1 Detailed Description	2704
4.581.2 Member Function Documentation	2704
4.582 std::ios_base Class Reference	2704
4.582.1 Detailed Description	2707
4.582.2 Member Typedef Documentation	2707
4.582.3 Member Enumeration Documentation	2709
4.582.4 Constructor & Destructor Documentation	2710
4.582.5 Member Function Documentation	2710
4.582.6 Member Data Documentation	2717
4.583 std::is_abstract< _Tp > Struct Template Reference	2724
4.583.1 Detailed Description	2725
4.584 std::is_arithmetic< _Tp > Struct Template Reference	2725
4.584.1 Detailed Description	2725
4.585 std::is_array< typename > Struct Template Reference	2726
4.585.1 Detailed Description	2726
4.586 std::is_assignable< _Tp, _Up > Struct Template Reference	2727
4.586.1 Detailed Description	2727
4.587 std::is_base_of< _Base, _Derived > Struct Template Reference	2728
4.587.1 Detailed Description	2728
4.588 std::is_bind_expression< _Tp > Struct Template Reference	2729
4.588.1 Detailed Description	2729
4.589 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference	2730
4.589.1 Detailed Description	2730
4.590 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference	2731
4.590.1 Detailed Description	2731
4.591 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference	2732
4.591.1 Detailed Description	2732

4.592 <code>std::is_bind_expression< const _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2733
4.592.1 Detailed Description	2733
4.593 <code>std::is_bind_expression< const volatile _Bind< _Signature > ></code> Struct Template Reference	2734
4.593.1 Detailed Description	2734
4.594 <code>std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2735
4.594.1 Detailed Description	2735
4.595 <code>std::is_bind_expression< volatile _Bind< _Signature > ></code> Struct Template Reference	2736
4.595.1 Detailed Description	2736
4.596 <code>std::is_bind_expression< volatile _Bind_result< _Result, _Signature > ></code> Struct Template Reference	2737
4.596.1 Detailed Description	2737
4.597 <code>std::is_class< _Tp ></code> Struct Template Reference	2738
4.597.1 Detailed Description	2738
4.598 <code>std::is_compound< _Tp ></code> Struct Template Reference	2739
4.598.1 Detailed Description	2739
4.599 <code>std::is_const< typename ></code> Struct Template Reference	2740
4.599.1 Detailed Description	2740
4.600 <code>std::is_constructible< _Tp, _Args ></code> Struct Template Reference	2741
4.600.1 Detailed Description	2741
4.601 <code>std::is_convertible< _From, _To ></code> Struct Template Reference	2741
4.601.1 Detailed Description	2741
4.602 <code>std::is_copy_assignable< _Tp ></code> Struct Template Reference	2742
4.602.1 Detailed Description	2742
4.603 <code>std::is_copy_constructible< _Tp ></code> Struct Template Reference	2742
4.603.1 Detailed Description	2742
4.604 <code>std::is_default_constructible< _Tp ></code> Struct Template Reference	2742
4.604.1 Detailed Description	2743
4.605 <code>std::is_destructible< _Tp ></code> Struct Template Reference	2743
4.605.1 Detailed Description	2743
4.606 <code>std::is_empty< _Tp ></code> Struct Template Reference	2743
4.606.1 Detailed Description	2744
4.607 <code>std::is_enum< _Tp ></code> Struct Template Reference	2744
4.607.1 Detailed Description	2745
4.608 <code>std::is_error_code_enum< _Tp ></code> Struct Template Reference	2745
4.608.1 Detailed Description	2746
4.609 <code>std::is_error_code_enum< future_errc ></code> Struct Template Reference	2746
4.609.1 Detailed Description	2747
4.610 <code>std::is_error_condition_enum< _Tp ></code> Struct Template Reference	2747
4.610.1 Detailed Description	2748

4.611	std::is_final< _Tp > Struct Template Reference	2748
4.611.1	Detailed Description	2749
4.612	std::is_floating_point< _Tp > Struct Template Reference	2749
4.612.1	Detailed Description	2749
4.613	std::is_function< _Tp > Struct Template Reference	2750
4.613.1	Detailed Description	2750
4.614	std::is_fundamental< _Tp > Struct Template Reference	2751
4.614.1	Detailed Description	2751
4.615	std::is_integral< _Tp > Struct Template Reference	2751
4.615.1	Detailed Description	2752
4.616	std::is_literal_type< _Tp > Struct Template Reference	2752
4.616.1	Detailed Description	2753
4.617	std::is_lvalue_reference< typename > Struct Template Reference	2753
4.617.1	Detailed Description	2754
4.618	std::is_member_function_pointer< _Tp > Struct Template Reference	2754
4.618.1	Detailed Description	2755
4.619	std::is_member_object_pointer< _Tp > Struct Template Reference	2755
4.619.1	Detailed Description	2755
4.620	std::is_member_pointer< _Tp > Struct Template Reference	2755
4.620.1	Detailed Description	2756
4.621	std::is_move_assignable< _Tp > Struct Template Reference	2756
4.621.1	Detailed Description	2756
4.622	std::is_move_constructible< _Tp > Struct Template Reference	2757
4.622.1	Detailed Description	2757
4.623	std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference	2757
4.623.1	Detailed Description	2757
4.624	std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference	2758
4.624.1	Detailed Description	2758
4.625	std::is_nothrow_copy_assignable< _Tp > Struct Template Reference	2759
4.625.1	Detailed Description	2759
4.626	std::is_nothrow_copy_constructible< _Tp > Struct Template Reference	2759
4.626.1	Detailed Description	2759
4.627	std::is_nothrow_default_constructible< _Tp > Struct Template Reference	2759
4.627.1	Detailed Description	2760
4.628	std::is_nothrow_destructible< _Tp > Struct Template Reference	2760
4.628.1	Detailed Description	2760
4.629	std::is_nothrow_move_assignable< _Tp > Struct Template Reference	2761
4.629.1	Detailed Description	2761
4.630	std::is_nothrow_move_constructible< _Tp > Struct Template Reference	2761

4.630.1 Detailed Description	2761
4.631 std::is_nothrow_swappable< _Tp > Struct Template Reference	2761
4.631.1 Detailed Description	2762
4.632 std::is_nothrow_swappable_with< _Tp, _Up > Struct Template Reference	2762
4.632.1 Detailed Description	2762
4.633 std::is_null_pointer< _Tp > Struct Template Reference	2762
4.633.1 Detailed Description	2763
4.634 std::is_object< _Tp > Struct Template Reference	2763
4.634.1 Detailed Description	2764
4.635 std::is_placeholder< _Tp > Struct Template Reference	2764
4.635.1 Detailed Description	2765
4.636 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference	2765
4.636.1 Detailed Description	2766
4.637 std::is_pod< _Tp > Struct Template Reference	2766
4.637.1 Detailed Description	2767
4.638 std::is_pointer< _Tp > Struct Template Reference	2767
4.638.1 Detailed Description	2767
4.639 std::is_polymorphic< _Tp > Struct Template Reference	2767
4.639.1 Detailed Description	2768
4.640 std::is_reference< _Tp > Struct Template Reference	2768
4.640.1 Detailed Description	2768
4.641 std::is_rvalue_reference< typename > Struct Template Reference	2769
4.641.1 Detailed Description	2769
4.642 std::is_same< _Tp, _Up > Struct Template Reference	2770
4.642.1 Detailed Description	2770
4.643 std::is_scalar< _Tp > Struct Template Reference	2771
4.643.1 Detailed Description	2771
4.644 std::is_standard_layout< _Tp > Struct Template Reference	2771
4.644.1 Detailed Description	2772
4.645 std::is_swappable< _Tp > Struct Template Reference	2772
4.645.1 Detailed Description	2772
4.646 std::is_swappable_with< _Tp, _Up > Struct Template Reference	2772
4.646.1 Detailed Description	2773
4.647 std::is_trivial< _Tp > Struct Template Reference	2773
4.647.1 Detailed Description	2774
4.648 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference	2774
4.648.1 Detailed Description	2775
4.649 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference	2775
4.649.1 Detailed Description	2776

4.650 std::is_trivially_copy_assignable<_Tp> Struct Template Reference	2776
4.650.1 Detailed Description	2776
4.651 std::is_trivially_copy_constructible<_Tp> Struct Template Reference	2776
4.651.1 Detailed Description	2776
4.652 std::is_trivially_default_constructible<_Tp> Struct Template Reference	2777
4.652.1 Detailed Description	2777
4.653 std::is_trivially_destructible<_Tp> Struct Template Reference	2778
4.653.1 Detailed Description	2778
4.654 std::is_trivially_move_assignable<_Tp> Struct Template Reference	2778
4.654.1 Detailed Description	2778
4.655 std::is_trivially_move_constructible<_Tp> Struct Template Reference	2778
4.655.1 Detailed Description	2779
4.656 std::is_union<_Tp> Struct Template Reference	2779
4.656.1 Detailed Description	2780
4.657 std::is_void<_Tp> Struct Template Reference	2780
4.657.1 Detailed Description	2781
4.658 std::is_volatile<typename> Struct Template Reference	2781
4.658.1 Detailed Description	2782
4.659 std::istream_iterator<_Tp, _CharT, _Traits, _Dist> Class Template Reference	2782
4.659.1 Detailed Description	2783
4.659.2 Member Typedef Documentation	2783
4.659.3 Constructor & Destructor Documentation	2784
4.659.4 Friends And Related Function Documentation	2785
4.660 std::istreambuf_iterator<_CharT, _Traits> Class Template Reference	2786
4.660.1 Detailed Description	2787
4.660.2 Member Typedef Documentation	2787
4.660.3 Constructor & Destructor Documentation	2789
4.660.4 Member Function Documentation	2790
4.661 std::experimental::filesystem::v1::path::iterator Class Reference	2791
4.661.1 Detailed Description	2792
4.662 __gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::iterator Struct Reference	2792
4.662.1 Detailed Description	2793
4.663 std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference> Struct Template Reference	2793
4.663.1 Detailed Description	2794
4.663.2 Member Typedef Documentation	2794
4.664 std::iterator_traits<_Iterator> Struct Template Reference	2795
4.664.1 Detailed Description	2796
4.665 std::iterator_traits<_Tp*> Struct Template Reference	2796
4.665.1 Detailed Description	2796

4.666 std::iterator_traits< const _Tp * > Struct Template Reference	2796
4.666.1 Detailed Description	2797
4.667 __gnu_pbds::join_error Struct Reference	2797
4.667.1 Detailed Description	2797
4.667.2 Member Function Documentation	2798
4.668 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference	2798
4.668.1 Detailed Description	2800
4.669 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference	2800
4.669.1 Detailed Description	2801
4.669.2 Member Typedef Documentation	2801
4.669.3 Constructor & Destructor Documentation	2802
4.669.4 Member Function Documentation	2803
4.670 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference	2805
4.670.1 Detailed Description	2805
4.671 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference	2806
4.671.1 Detailed Description	2807
4.671.2 Member Typedef Documentation	2807
4.671.3 Constructor & Destructor Documentation	2808
4.671.4 Member Function Documentation	2809
4.672 std::length_error Class Reference	2810
4.672.1 Detailed Description	2811
4.672.2 Member Function Documentation	2811
4.673 std::less< _Tp > Struct Template Reference	2811
4.673.1 Detailed Description	2812
4.673.2 Member Typedef Documentation	2812
4.674 std::less< void > Struct Template Reference	2813
4.674.1 Detailed Description	2813
4.675 std::less_equal< _Tp > Struct Template Reference	2814
4.675.1 Detailed Description	2814
4.675.2 Member Typedef Documentation	2814
4.676 std::less_equal< void > Struct Template Reference	2815
4.676.1 Detailed Description	2816
4.677 __gnu_cxx::limit_condition::limit_adjustor Struct Reference	2816
4.677.1 Detailed Description	2816
4.678 __gnu_cxx::limit_condition Struct Reference	2816
4.678.1 Detailed Description	2817

4.679	std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference	2817
4.679.1	Detailed Description	2818
4.679.2	Member Typedef Documentation	2818
4.679.3	Constructor & Destructor Documentation	2819
4.679.4	Member Function Documentation	2820
4.679.5	Friends And Related Function Documentation	2822
4.679.6	Member Data Documentation	2823
4.680	__gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference	2824
4.680.1	Detailed Description	2824
4.680.2	Member Function Documentation	2825
4.681	std::__debug::list< _Tp, _Allocator > Class Template Reference	2825
4.681.1	Detailed Description	2828
4.681.2	Member Function Documentation	2828
4.681.3	Member Data Documentation	2830
4.682	std::list< _Tp, _Alloc > Class Template Reference	2831
4.682.1	Detailed Description	2834
4.682.2	Constructor & Destructor Documentation	2835
4.682.3	Member Function Documentation	2838
4.683	__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference	2859
4.683.1	Detailed Description	2859
4.683.2	Constructor & Destructor Documentation	2860
4.684	__gnu_pbds::list_update_tag Struct Reference	2861
4.684.1	Detailed Description	2861
4.685	std::locale Class Reference	2861
4.685.1	Detailed Description	2863
4.685.2	Member Typedef Documentation	2863
4.685.3	Constructor & Destructor Documentation	2863
4.685.4	Member Function Documentation	2867
4.685.5	Friends And Related Function Documentation	2870
4.685.6	Member Data Documentation	2871
4.686	std::lock_guard< _Mutex > Class Template Reference	2874
4.686.1	Detailed Description	2874
4.687	std::logic_error Class Reference	2874
4.687.1	Detailed Description	2875
4.687.2	Constructor & Destructor Documentation	2875
4.687.3	Member Function Documentation	2875
4.688	std::logical_and< _Tp > Struct Template Reference	2876
4.688.1	Detailed Description	2876
4.688.2	Member Typedef Documentation	2876

4.689	std::logical_and< void > Struct Template Reference	2877
4.689.1	Detailed Description	2878
4.690	std::logical_not< _Tp > Struct Template Reference	2878
4.690.1	Detailed Description	2878
4.690.2	Member Typedef Documentation	2879
4.691	std::logical_not< void > Struct Template Reference	2879
4.691.1	Detailed Description	2879
4.692	std::logical_or< _Tp > Struct Template Reference	2880
4.692.1	Detailed Description	2880
4.692.2	Member Typedef Documentation	2880
4.693	std::logical_or< void > Struct Template Reference	2881
4.693.1	Detailed Description	2882
4.694	std::lognormal_distribution< _RealType > Class Template Reference	2882
4.694.1	Detailed Description	2883
4.694.2	Member Typedef Documentation	2883
4.694.3	Member Function Documentation	2883
4.694.4	Friends And Related Function Documentation	2885
4.695	__gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference	2886
4.695.1	Detailed Description	2887
4.696	__gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference	2887
4.696.1	Detailed Description	2888
4.696.2	Member Typedef Documentation	2888
4.696.3	Member Enumeration Documentation	2888
4.696.4	Member Function Documentation	2889
4.697	__gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference	2889
4.697.1	Detailed Description	2890
4.698	__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference	2890
4.698.1	Detailed Description	2892
4.699	__gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference	2892
4.699.1	Detailed Description	2893
4.699.2	Member Typedef Documentation	2893
4.699.3	Member Function Documentation	2893
4.700	std::make_signed< _Tp > Struct Template Reference	2894
4.700.1	Detailed Description	2894
4.701	std::make_unsigned< _Tp > Struct Template Reference	2894
4.701.1	Detailed Description	2895
4.702	__gnu_cxx::malloc_allocator< _Tp > Class Template Reference	2895
4.702.1	Detailed Description	2896
4.703	std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference	2896

4.703.1 Detailed Description	2899
4.703.2 Member Function Documentation	2899
4.703.3 Member Data Documentation	2901
4.704 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference	2902
4.704.1 Detailed Description	2905
4.704.2 Constructor & Destructor Documentation	2905
4.704.3 Member Function Documentation	2909
4.705 std::mask_array< _Tp > Class Template Reference	2934
4.705.1 Detailed Description	2935
4.706 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference	2936
4.706.1 Detailed Description	2936
4.707 std::match_results< _Bi_iter, _Alloc > Class Template Reference	2937
4.707.1 Detailed Description	2941
4.707.2 Constructor & Destructor Documentation	2941
4.707.3 Member Function Documentation	2942
4.708 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	2950
4.708.1 Detailed Description	2950
4.709 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference	2950
4.709.1 Detailed Description	2951
4.710 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference	2951
4.710.1 Detailed Description	2952
4.710.2 Member Typedef Documentation	2952
4.711 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference	2953
4.711.1 Detailed Description	2953
4.711.2 Member Typedef Documentation	2953
4.712 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference	2954
4.712.1 Detailed Description	2955
4.712.2 Member Typedef Documentation	2955
4.713 std::mem_fun_t< _Ret, _Tp > Class Template Reference	2956
4.713.1 Detailed Description	2956
4.713.2 Member Typedef Documentation	2956
4.714 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference	2957
4.714.1 Detailed Description	2958
4.714.2 Member Typedef Documentation	2959
4.714.3 Constructor & Destructor Documentation	2960
4.714.4 Member Function Documentation	2960
4.714.5 Friends And Related Function Documentation	2961
4.715 std::messages< _CharT > Class Template Reference	2963

4.715.1 Detailed Description	2964
4.715.2 Member Typedef Documentation	2965
4.715.3 Constructor & Destructor Documentation	2965
4.715.4 Member Function Documentation	2966
4.715.5 Member Data Documentation	2966
4.716 std::messages_base Struct Reference	2967
4.716.1 Detailed Description	2967
4.717 std::messages_byname< _CharT > Class Template Reference	2968
4.717.1 Detailed Description	2969
4.717.2 Member Function Documentation	2969
4.717.3 Member Data Documentation	2970
4.718 std::minus< _Tp > Struct Template Reference	2970
4.718.1 Detailed Description	2971
4.718.2 Member Typedef Documentation	2971
4.719 std::minus< void > Struct Template Reference	2972
4.719.1 Detailed Description	2972
4.720 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference	2973
4.720.1 Detailed Description	2973
4.721 std::modulus< _Tp > Struct Template Reference	2974
4.721.1 Detailed Description	2974
4.721.2 Member Typedef Documentation	2974
4.722 std::modulus< void > Struct Template Reference	2975
4.722.1 Detailed Description	2976
4.723 std::money_base Class Reference	2976
4.723.1 Detailed Description	2977
4.724 std::money_get< _CharT, _InIter > Class Template Reference	2977
4.724.1 Detailed Description	2978
4.724.2 Member Typedef Documentation	2979
4.724.3 Constructor & Destructor Documentation	2979
4.724.4 Member Function Documentation	2980
4.724.5 Member Data Documentation	2983
4.725 std::money_put< _CharT, _OutIter > Class Template Reference	2983
4.725.1 Detailed Description	2984
4.725.2 Member Typedef Documentation	2985
4.725.3 Constructor & Destructor Documentation	2985
4.725.4 Member Function Documentation	2986
4.725.5 Member Data Documentation	2989
4.726 std::moneypunct< _CharT, _Intl > Class Template Reference	2990
4.726.1 Detailed Description	2991

4.726.2 Member Typedef Documentation	2992
4.726.3 Constructor & Destructor Documentation	2992
4.726.4 Member Function Documentation	2994
4.726.5 Member Data Documentation	3002
4.727 std::moneypunct_byname< _CharT, _Intl > Class Template Reference	3003
4.727.1 Detailed Description	3005
4.727.2 Member Function Documentation	3005
4.727.3 Member Data Documentation	3014
4.728 std::move_iterator< _Iterator > Class Template Reference	3014
4.728.1 Detailed Description	3015
4.729 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference	3015
4.729.1 Detailed Description	3018
4.729.2 Member Function Documentation	3018
4.729.3 Member Data Documentation	3020
4.730 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference	3021
4.730.1 Detailed Description	3024
4.730.2 Constructor & Destructor Documentation	3025
4.730.3 Member Function Documentation	3029
4.731 std::multiplies< _Tp > Struct Template Reference	3052
4.731.1 Detailed Description	3052
4.731.2 Member Typedef Documentation	3052
4.732 std::multiplies< void > Struct Template Reference	3053
4.732.1 Detailed Description	3054
4.733 std::multiset< _Key, _Compare, _Alloc > Class Template Reference	3054
4.733.1 Detailed Description	3056
4.733.2 Constructor & Destructor Documentation	3057
4.733.3 Member Function Documentation	3061
4.734 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference	3080
4.734.1 Detailed Description	3083
4.734.2 Member Function Documentation	3083
4.734.3 Member Data Documentation	3085
4.735 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference	3086
4.735.1 Detailed Description	3086
4.735.2 Member Function Documentation	3086
4.736 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference	3087
4.736.1 Detailed Description	3087
4.736.2 Member Function Documentation	3087
4.737 __gnu_parallel::multiway_mergesort_tag Struct Reference	3088
4.737.1 Detailed Description	3089

4.737.2 Member Function Documentation	3089
4.738 std::mutex Class Reference	3090
4.738.1 Detailed Description	3090
4.739 std::negate< _Tp > Struct Template Reference	3091
4.739.1 Detailed Description	3091
4.739.2 Member Typedef Documentation	3091
4.740 std::negate< void > Struct Template Reference	3092
4.740.1 Detailed Description	3092
4.741 std::negative_binomial_distribution< _IntType > Class Template Reference	3092
4.741.1 Detailed Description	3093
4.741.2 Member Typedef Documentation	3094
4.741.3 Member Function Documentation	3094
4.741.4 Friends And Related Function Documentation	3096
4.742 std::nested_exception Class Reference	3097
4.742.1 Detailed Description	3097
4.743 __gnu_cxx::limit_condition::never_adjustor Struct Reference	3098
4.743.1 Detailed Description	3098
4.744 __gnu_cxx::random_condition::never_adjustor Struct Reference	3098
4.744.1 Detailed Description	3098
4.745 __gnu_cxx::new_allocator< _Tp > Class Template Reference	3098
4.745.1 Detailed Description	3099
4.746 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference	3100
4.746.1 Detailed Description	3100
4.747 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference	3100
4.747.1 Detailed Description	3101
4.748 std::normal_distribution< _RealType > Class Template Reference	3101
4.748.1 Detailed Description	3102
4.748.2 Member Typedef Documentation	3102
4.748.3 Constructor & Destructor Documentation	3102
4.748.4 Member Function Documentation	3103
4.748.5 Friends And Related Function Documentation	3105
4.749 std::not_equal_to< _Tp > Struct Template Reference	3107
4.749.1 Detailed Description	3107
4.749.2 Member Typedef Documentation	3107
4.750 std::not_equal_to< void > Struct Template Reference	3108
4.750.1 Detailed Description	3109
4.751 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference	3109
4.751.1 Detailed Description	3109
4.752 __gnu_pbds::null_type Struct Reference	3110

4.752.1 Detailed Description	3110
4.753 std::experimental::fundamentals_v1::nullopt_t Struct Reference	3110
4.753.1 Detailed Description	3111
4.754 std::num_get<_CharT, _InIter> Class Template Reference	3111
4.754.1 Detailed Description	3113
4.754.2 Member Typedef Documentation	3113
4.754.3 Constructor & Destructor Documentation	3114
4.754.4 Member Function Documentation	3114
4.754.5 Member Data Documentation	3132
4.755 std::num_put<_CharT, _OutIter> Class Template Reference	3133
4.755.1 Detailed Description	3134
4.755.2 Member Typedef Documentation	3135
4.755.3 Constructor & Destructor Documentation	3135
4.755.4 Member Function Documentation	3136
4.755.5 Member Data Documentation	3149
4.756 std::numeric_limits<_Tp> Struct Template Reference	3149
4.756.1 Detailed Description	3150
4.756.2 Member Function Documentation	3151
4.756.3 Member Data Documentation	3153
4.757 std::numeric_limits<bool> Struct Template Reference	3158
4.757.1 Detailed Description	3158
4.758 std::numeric_limits<char> Struct Template Reference	3159
4.758.1 Detailed Description	3159
4.759 std::numeric_limits<char16_t> Struct Template Reference	3160
4.759.1 Detailed Description	3160
4.760 std::numeric_limits<char32_t> Struct Template Reference	3161
4.760.1 Detailed Description	3161
4.761 std::numeric_limits<double> Struct Template Reference	3162
4.761.1 Detailed Description	3162
4.762 std::numeric_limits<float> Struct Template Reference	3163
4.762.1 Detailed Description	3163
4.763 std::numeric_limits<int> Struct Template Reference	3164
4.763.1 Detailed Description	3164
4.764 std::numeric_limits<long> Struct Template Reference	3165
4.764.1 Detailed Description	3165
4.765 std::numeric_limits<long double> Struct Template Reference	3166
4.765.1 Detailed Description	3166
4.766 std::numeric_limits<long long> Struct Template Reference	3167
4.766.1 Detailed Description	3167

4.767 std::numeric_limits< short > Struct Template Reference	3168
4.767.1 Detailed Description	3168
4.768 std::numeric_limits< signed char > Struct Template Reference	3169
4.768.1 Detailed Description	3169
4.769 std::numeric_limits< unsigned char > Struct Template Reference	3170
4.769.1 Detailed Description	3170
4.770 std::numeric_limits< unsigned int > Struct Template Reference	3171
4.770.1 Detailed Description	3171
4.771 std::numeric_limits< unsigned long > Struct Template Reference	3172
4.771.1 Detailed Description	3172
4.772 std::numeric_limits< unsigned long long > Struct Template Reference	3173
4.772.1 Detailed Description	3173
4.773 std::numeric_limits< unsigned short > Struct Template Reference	3174
4.773.1 Detailed Description	3174
4.774 std::numeric_limits< wchar_t > Struct Template Reference	3175
4.774.1 Detailed Description	3175
4.775 std::numpunct< _CharT > Class Template Reference	3176
4.775.1 Detailed Description	3177
4.775.2 Member Typedef Documentation	3177
4.775.3 Constructor & Destructor Documentation	3178
4.775.4 Member Function Documentation	3179
4.775.5 Member Data Documentation	3184
4.776 std::numpunct_byname< _CharT > Class Template Reference	3184
4.776.1 Detailed Description	3186
4.776.2 Member Function Documentation	3186
4.776.3 Member Data Documentation	3190
4.777 __gnu_parallel::omp_loop_static_tag Struct Reference	3190
4.777.1 Detailed Description	3191
4.777.2 Member Function Documentation	3191
4.778 __gnu_parallel::omp_loop_tag Struct Reference	3192
4.778.1 Detailed Description	3192
4.778.2 Member Function Documentation	3192
4.779 std::once_flag Struct Reference	3193
4.779.1 Detailed Description	3193
4.779.2 Constructor & Destructor Documentation	3193
4.779.3 Member Function Documentation	3194
4.779.4 Friends And Related Function Documentation	3194
4.780 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference	3194
4.780.1 Detailed Description	3196

4.781	std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference	3196
4.781.1	Detailed Description	3197
4.781.2	Member Typedef Documentation	3197
4.781.3	Constructor & Destructor Documentation	3199
4.781.4	Member Function Documentation	3200
4.782	std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference	3200
4.782.1	Detailed Description	3201
4.783	std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference	3201
4.783.1	Detailed Description	3202
4.783.2	Member Typedef Documentation	3202
4.783.3	Constructor & Destructor Documentation	3204
4.783.4	Member Function Documentation	3205
4.784	std::out_of_range Class Reference	3207
4.784.1	Detailed Description	3207
4.784.2	Member Function Documentation	3207
4.785	std::output_iterator_tag Struct Reference	3208
4.785.1	Detailed Description	3208
4.786	__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	3208
4.786.1	Detailed Description	3210
4.786.2	Member Function Documentation	3210
4.787	__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference	3212
4.787.1	Detailed Description	3213
4.787.2	Member Function Documentation	3213
4.788	__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference	3214
4.788.1	Detailed Description	3215
4.788.2	Member Function Documentation	3215
4.789	__gnu_pbds::ov_tree_tag Struct Reference	3216
4.789.1	Detailed Description	3216
4.790	std::overflow_error Class Reference	3217
4.790.1	Detailed Description	3217
4.790.2	Member Function Documentation	3217
4.791	std::owner_less< _Tp > Struct Template Reference	3218
4.791.1	Detailed Description	3218
4.792	std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference	3218
4.792.1	Detailed Description	3219
4.792.2	Member Typedef Documentation	3219
4.793	std::owner_less< shared_ptr< _Tp > > Struct Template Reference	3219

4.793.1 Detailed Description	3220
4.793.2 Member Typedef Documentation	3220
4.794 std::owner_less< void > Struct Template Reference	3221
4.794.1 Detailed Description	3221
4.794.2 Member Typedef Documentation	3221
4.795 std::owner_less< weak_ptr< _Tp > > Struct Template Reference	3222
4.795.1 Detailed Description	3223
4.795.2 Member Typedef Documentation	3223
4.796 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference	3223
4.796.1 Detailed Description	3224
4.796.2 Member Typedef Documentation	3224
4.797 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference	3225
4.797.1 Detailed Description	3225
4.798 std::pair< _T1, _T2 > Struct Template Reference	3226
4.798.1 Detailed Description	3227
4.798.2 Member Typedef Documentation	3228
4.798.3 Constructor & Destructor Documentation	3228
4.798.4 Member Function Documentation	3230
4.798.5 Member Data Documentation	3230
4.799 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference	3231
4.799.1 Detailed Description	3233
4.800 __gnu_pbds::pairing_heap_tag Struct Reference	3233
4.800.1 Detailed Description	3233
4.801 __gnu_parallel::parallel_tag Struct Reference	3234
4.801.1 Detailed Description	3235
4.801.2 Constructor & Destructor Documentation	3235
4.801.3 Member Function Documentation	3235
4.802 std::poisson_distribution< _IntType >::param_type Struct Reference	3236
4.802.1 Detailed Description	3236
4.803 std::chi_squared_distribution< _RealType >::param_type Struct Reference	3237
4.803.1 Detailed Description	3237
4.804 std::binomial_distribution< _IntType >::param_type Struct Reference	3237
4.804.1 Detailed Description	3238
4.805 std::lognormal_distribution< _RealType >::param_type Struct Reference	3238
4.805.1 Detailed Description	3238
4.806 std::student_t_distribution< _RealType >::param_type Struct Reference	3239
4.806.1 Detailed Description	3239
4.807 std::extreme_value_distribution< _RealType >::param_type Struct Reference	3239
4.807.1 Detailed Description	3240

4.808 std::discrete_distribution< _IntType >::param_type Struct Reference	3240
4.808.1 Detailed Description	3240
4.809 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference	3241
4.809.1 Detailed Description	3241
4.810 std::fisher_f_distribution< _RealType >::param_type Struct Reference	3241
4.810.1 Detailed Description	3242
4.811 std::bernoulli_distribution::param_type Struct Reference	3242
4.811.1 Detailed Description	3243
4.812 std::exponential_distribution< _RealType >::param_type Struct Reference	3243
4.812.1 Detailed Description	3243
4.813 std::weibull_distribution< _RealType >::param_type Struct Reference	3243
4.813.1 Detailed Description	3244
4.814 std::normal_distribution< _RealType >::param_type Struct Reference	3244
4.814.1 Detailed Description	3245
4.815 std::negative_binomial_distribution< _IntType >::param_type Struct Reference	3245
4.815.1 Detailed Description	3245
4.816 std::geometric_distribution< _IntType >::param_type Struct Reference	3246
4.816.1 Detailed Description	3246
4.817 std::gamma_distribution< _RealType >::param_type Struct Reference	3246
4.817.1 Detailed Description	3247
4.818 std::cauchy_distribution< _RealType >::param_type Struct Reference	3247
4.818.1 Detailed Description	3247
4.819 std::uniform_real_distribution< _RealType >::param_type Struct Reference	3248
4.819.1 Detailed Description	3248
4.820 std::uniform_int_distribution< _IntType >::param_type Struct Reference	3248
4.820.1 Detailed Description	3249
4.821 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference	3249
4.821.1 Detailed Description	3250
4.822 __gnu_pbds::detail::pat_trie_base Struct Reference	3250
4.822.1 Detailed Description	3251
4.822.2 Member Enumeration Documentation	3251
4.823 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference	3251
4.823.1 Detailed Description	3253
4.823.2 Member Enumeration Documentation	3254
4.823.3 Member Function Documentation	3254
4.824 __gnu_pbds::pat_trie_tag Struct Reference	3255
4.824.1 Detailed Description	3255
4.825 std::experimental::filesystem::v1::path Class Reference	3256

4.825.1 Detailed Description	3258
4.826 std::piecewise_constant_distribution< _RealType > Class Template Reference	3258
4.826.1 Detailed Description	3259
4.826.2 Member Typedef Documentation	3259
4.826.3 Member Function Documentation	3260
4.826.4 Friends And Related Function Documentation	3262
4.827 std::piecewise_construct_t Struct Reference	3263
4.827.1 Detailed Description	3263
4.828 std::piecewise_linear_distribution< _RealType > Class Template Reference	3263
4.828.1 Detailed Description	3264
4.828.2 Member Typedef Documentation	3265
4.828.3 Member Function Documentation	3265
4.828.4 Friends And Related Function Documentation	3267
4.829 std::plus< _Tp > Struct Template Reference	3269
4.829.1 Detailed Description	3269
4.829.2 Member Typedef Documentation	3269
4.830 __gnu_pbds::point_invalidation_guarantee Struct Reference	3270
4.830.1 Detailed Description	3271
4.831 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference	3271
4.831.1 Detailed Description	3272
4.831.2 Member Typedef Documentation	3272
4.832 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference	3273
4.832.1 Detailed Description	3273
4.832.2 Member Typedef Documentation	3273
4.833 std::pointer_traits< _Ptr > Struct Template Reference	3274
4.833.1 Detailed Description	3274
4.833.2 Member Typedef Documentation	3275
4.834 std::pointer_traits< _Tp * > Struct Template Reference	3276
4.834.1 Detailed Description	3276
4.834.2 Member Typedef Documentation	3276
4.834.3 Member Function Documentation	3277
4.835 std::poisson_distribution< _IntType > Class Template Reference	3277
4.835.1 Detailed Description	3278
4.835.2 Member Typedef Documentation	3279
4.835.3 Member Function Documentation	3279
4.835.4 Friends And Related Function Documentation	3281
4.836 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference	3282
4.836.1 Detailed Description	3283
4.837 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference	3284

4.837.1 Detailed Description	3285
4.837.2 Constructor & Destructor Documentation	3285
4.837.3 Member Function Documentation	3286
4.838 <code>__gnu_pbds::priority_queue_tag</code> Struct Reference	3288
4.838.1 Detailed Description	3289
4.839 <code>__gnu_pbds::detail::probe_fn_base<_Alloc></code> Class Template Reference	3289
4.839.1 Detailed Description	3289
4.840 <code>__gnu_cxx::project1st<_Arg1, _Arg2></code> Struct Template Reference	3289
4.840.1 Detailed Description	3290
4.840.2 Member Typedef Documentation	3290
4.841 <code>__gnu_cxx::project2nd<_Arg1, _Arg2></code> Struct Template Reference	3290
4.841.1 Detailed Description	3291
4.841.2 Member Typedef Documentation	3291
4.842 <code>std::promise<_Res></code> Class Template Reference	3292
4.842.1 Detailed Description	3292
4.843 <code>std::promise<_Res &></code> Class Template Reference	3293
4.843.1 Detailed Description	3293
4.844 <code>std::promise<void></code> Class Template Reference	3293
4.844.1 Detailed Description	3294
4.845 <code>std::experimental::fundamentals_v2::propagate_const<_Tp></code> Class Template Reference	3294
4.845.1 Detailed Description	3295
4.846 <code>__gnu_pbds::quadratic_probe_fn<Size_Type></code> Class Template Reference	3295
4.846.1 Detailed Description	3296
4.846.2 Member Function Documentation	3296
4.847 <code>std::queue<_Tp, _Sequence></code> Class Template Reference	3296
4.847.1 Detailed Description	3297
4.847.2 Constructor & Destructor Documentation	3298
4.847.3 Member Function Documentation	3298
4.847.4 Member Data Documentation	3301
4.848 <code>__gnu_parallel::quicksort_tag</code> Struct Reference	3302
4.848.1 Detailed Description	3302
4.848.2 Member Function Documentation	3302
4.849 <code>std::random_access_iterator_tag</code> Struct Reference	3303
4.849.1 Detailed Description	3303
4.850 <code>__gnu_cxx::random_condition</code> Struct Reference	3304
4.850.1 Detailed Description	3304
4.851 <code>std::random_device</code> Class Reference	3304
4.851.1 Detailed Description	3305
4.851.2 Member Typedef Documentation	3305

4.852 <code>std::range_error</code> Class Reference	3306
4.852.1 Detailed Description	3306
4.852.2 Member Function Documentation	3306
4.853 <code>__gnu_pbds::range_invalidation_guarantee</code> Struct Reference	3307
4.853.1 Detailed Description	3307
4.854 <code>__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference</code>	3308
4.854.1 Detailed Description	3308
4.855 <code>__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference</code>	3308
4.855.1 Detailed Description	3309
4.856 <code>__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference</code>	3309
4.856.1 Detailed Description	3310
4.857 <code>__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference</code>	3310
4.857.1 Detailed Description	3310
4.858 <code>__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference</code>	3311
4.858.1 Detailed Description	3311
4.859 <code>__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference</code>	3311
4.859.1 Detailed Description	3312
4.860 <code>__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference</code>	3312
4.860.1 Detailed Description	3312
4.861 <code>__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference</code>	3313
4.861.1 Detailed Description	3313
4.862 <code>__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference</code>	3313
4.862.1 Detailed Description	3314
4.863 <code>std::rank< typename > Struct Template Reference</code>	3314
4.863.1 Detailed Description	3315
4.864 <code>std::ratio< _Num, _Den > Struct Template Reference</code>	3315
4.864.1 Detailed Description	3316
4.865 <code>std::ratio_equal< _R1, _R2 > Struct Template Reference</code>	3316
4.865.1 Detailed Description	3317
4.866 <code>std::ratio_greater< _R1, _R2 > Struct Template Reference</code>	3317
4.866.1 Detailed Description	3318
4.867 <code>std::ratio_greater_equal< _R1, _R2 > Struct Template Reference</code>	3318
4.867.1 Detailed Description	3319

4.868 <code>std::ratio_less<_R1, _R2></code> Struct Template Reference	3319
4.868.1 Detailed Description	3319
4.869 <code>std::ratio_less_equal<_R1, _R2></code> Struct Template Reference	3319
4.869.1 Detailed Description	3320
4.870 <code>std::ratio_not_equal<_R1, _R2></code> Struct Template Reference	3320
4.870.1 Detailed Description	3321
4.871 <code>std::raw_storage_iterator<_OutputIterator, _Tp></code> Class Template Reference	3321
4.871.1 Detailed Description	3322
4.871.2 Member Typedef Documentation	3322
4.872 <code>__gnu_cxx::rb_tree<_Key, _Value, _KeyOfValue, _Compare, _Alloc></code> Struct Template Reference	3323
4.872.1 Detailed Description	3327
4.873 <code>__gnu_pbds::detail::rb_tree_map<_Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc></code> Class Template Reference	3327
4.873.1 Detailed Description	3330
4.873.2 Member Function Documentation	3331
4.874 <code>__gnu_pbds::detail::rb_tree_node<_Value_Type, Metadata, _Alloc></code> Struct Template Reference	3332
4.874.1 Detailed Description	3333
4.875 <code>__gnu_pbds::rb_tree_tag</code> Struct Reference	3333
4.875.1 Detailed Description	3333
4.876 <code>__gnu_pbds::detail::rc<_Node, _Alloc></code> Class Template Reference	3334
4.876.1 Detailed Description	3334
4.877 <code>__gnu_pbds::detail::rc_binomial_heap<_Value_Type, Cmp_Fn, _Alloc></code> Class Template Reference	3334
4.877.1 Detailed Description	3336
4.878 <code>__gnu_pbds::rc_binomial_heap_tag</code> Struct Reference	3337
4.878.1 Detailed Description	3337
4.879 <code>__gnu_pbds::detail::rebind_traits<_Alloc, T></code> Struct Template Reference	3337
4.879.1 Detailed Description	3338
4.880 <code>__gnu_cxx::recursive_init_error</code> Class Reference	3338
4.880.1 Detailed Description	3338
4.881 <code>std::recursive_mutex</code> Class Reference	3339
4.881.1 Detailed Description	3339
4.882 <code>std::recursive_timed_mutex</code> Class Reference	3339
4.882.1 Detailed Description	3340
4.883 <code>std::bitset<_Nb>::reference</code> Class Reference	3340
4.883.1 Detailed Description	3340
4.884 <code>std::tr2::dynamic_bitset<_WordT, _Alloc>::reference</code> Class Reference	3341
4.884.1 Detailed Description	3341
4.885 <code>std::reference_wrapper<_Tp></code> Class Template Reference	3341
4.885.1 Detailed Description	3342

4.885.2 Friends And Related Function Documentation	3342
4.886 std::regex_error Class Reference	3344
4.886.1 Detailed Description	3344
4.886.2 Constructor & Destructor Documentation	3344
4.886.3 Member Function Documentation	3345
4.887 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3345
4.887.1 Detailed Description	3346
4.887.2 Constructor & Destructor Documentation	3346
4.887.3 Member Function Documentation	3347
4.888 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference	3349
4.888.1 Detailed Description	3350
4.888.2 Constructor & Destructor Documentation	3350
4.888.3 Member Function Documentation	3353
4.889 std::regex_traits< _Ch_type > Class Template Reference	3355
4.889.1 Detailed Description	3355
4.889.2 Constructor & Destructor Documentation	3356
4.889.3 Member Function Documentation	3356
4.890 std::remove_all_extents< _Tp > Struct Template Reference	3363
4.890.1 Detailed Description	3363
4.891 std::remove_const< _Tp > Struct Template Reference	3363
4.891.1 Detailed Description	3364
4.892 std::remove_cv< _Tp > Struct Template Reference	3364
4.892.1 Detailed Description	3364
4.893 std::remove_extent< _Tp > Struct Template Reference	3364
4.893.1 Detailed Description	3365
4.894 std::remove_pointer< _Tp > Struct Template Reference	3365
4.894.1 Detailed Description	3365
4.895 std::remove_reference< _Tp > Struct Template Reference	3365
4.895.1 Detailed Description	3366
4.896 std::remove_volatile< _Tp > Struct Template Reference	3366
4.896.1 Detailed Description	3366
4.897 __gnu_pbds::resize_error Struct Reference	3367
4.897.1 Detailed Description	3367
4.897.2 Member Function Documentation	3367
4.898 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference	3368
4.898.1 Detailed Description	3368
4.899 std::result_of< _Signature > Class Template Reference	3369
4.899.1 Detailed Description	3369
4.900 std::reverse_iterator< _Iterator > Class Template Reference	3369

4.900.1 Detailed Description	3370
4.900.2 Member Typedef Documentation	3371
4.900.3 Constructor & Destructor Documentation	3371
4.900.4 Member Function Documentation	3372
4.901 <code>__gnu_cxx::rope<_CharT, _Alloc></code> Class Template Reference	3376
4.901.1 Detailed Description	3381
4.902 <code>std::runtime_error</code> Class Reference	3381
4.902.1 Detailed Description	3382
4.902.2 Constructor & Destructor Documentation	3382
4.902.3 Member Function Documentation	3382
4.903 <code>__gnu_pbds::sample_probe_fn</code> Class Reference	3383
4.903.1 Detailed Description	3383
4.903.2 Constructor & Destructor Documentation	3383
4.903.3 Member Function Documentation	3383
4.904 <code>__gnu_pbds::sample_range_hashing</code> Class Reference	3384
4.904.1 Detailed Description	3384
4.904.2 Member Typedef Documentation	3385
4.904.3 Constructor & Destructor Documentation	3385
4.904.4 Member Function Documentation	3385
4.905 <code>__gnu_pbds::sample_ranged_hash_fn</code> Class Reference	3386
4.905.1 Detailed Description	3386
4.905.2 Constructor & Destructor Documentation	3386
4.905.3 Member Function Documentation	3387
4.906 <code>__gnu_pbds::sample_ranged_probe_fn</code> Class Reference	3388
4.906.1 Detailed Description	3388
4.907 <code>__gnu_pbds::sample_resize_policy</code> Class Reference	3388
4.907.1 Detailed Description	3389
4.907.2 Member Typedef Documentation	3389
4.907.3 Constructor & Destructor Documentation	3389
4.907.4 Member Function Documentation	3390
4.908 <code>__gnu_pbds::sample_resize_trigger</code> Class Reference	3393
4.908.1 Detailed Description	3393
4.908.2 Member Typedef Documentation	3393
4.908.3 Constructor & Destructor Documentation	3394
4.908.4 Member Function Documentation	3394
4.909 <code>__gnu_pbds::sample_size_policy</code> Class Reference	3397
4.909.1 Detailed Description	3397
4.909.2 Member Typedef Documentation	3397
4.909.3 Constructor & Destructor Documentation	3398

4.909.4 Member Function Documentation	3398
4.910 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference	3399
4.910.1 Detailed Description	3399
4.911 __gnu_pbds::sample_trie_access_traits Struct Reference	3399
4.911.1 Detailed Description	3400
4.911.2 Member Typedef Documentation	3400
4.911.3 Member Function Documentation	3400
4.912 __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	3401
4.912.1 Detailed Description	3401
4.912.2 Constructor & Destructor Documentation	3401
4.912.3 Member Function Documentation	3401
4.913 __gnu_pbds::sample_update_policy Struct Reference	3402
4.913.1 Detailed Description	3402
4.913.2 Member Typedef Documentation	3402
4.913.3 Constructor & Destructor Documentation	3402
4.913.4 Member Function Documentation	3403
4.914 __gnu_parallel::sampling_tag Struct Reference	3404
4.914.1 Detailed Description	3404
4.914.2 Member Function Documentation	3404
4.915 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference	3405
4.915.1 Detailed Description	3407
4.916 std::seed_seq Class Reference	3407
4.916.1 Detailed Description	3407
4.916.2 Member Typedef Documentation	3407
4.916.3 Constructor & Destructor Documentation	3408
4.917 __gnu_cxx::select1st< _Pair > Struct Template Reference	3408
4.917.1 Detailed Description	3409
4.917.2 Member Typedef Documentation	3409
4.918 __gnu_cxx::select2nd< _Pair > Struct Template Reference	3409
4.918.1 Detailed Description	3410
4.918.2 Member Typedef Documentation	3410
4.919 __gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference	3410
4.919.1 Detailed Description	3411
4.920 __gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference	3411
4.920.1 Detailed Description	3411
4.921 std::basic_istream< _CharT, _Traits >::sentry Class Reference	3411
4.921.1 Detailed Description	3412
4.921.2 Member Typedef Documentation	3412

4.921.3 Constructor & Destructor Documentation	3412
4.921.4 Member Function Documentation	3413
4.922 <code>std::basic_ostream< _CharT, _Traits >::sentry</code> Class Reference	3413
4.922.1 Detailed Description	3414
4.922.2 Constructor & Destructor Documentation	3414
4.922.3 Member Function Documentation	3415
4.923 <code>__gnu_pbds::sequence_tag</code> Struct Reference	3415
4.923.1 Detailed Description	3416
4.924 <code>__gnu_parallel::sequential_tag</code> Struct Reference	3416
4.924.1 Detailed Description	3416
4.925 <code>std::set< _Key, _Compare, _Alloc ></code> Class Template Reference	3416
4.925.1 Detailed Description	3419
4.925.2 Member Typedef Documentation	3419
4.925.3 Constructor & Destructor Documentation	3423
4.925.4 Member Function Documentation	3427
4.926 <code>std::__debug::set< _Key, _Compare, _Allocator ></code> Class Template Reference	3446
4.926.1 Detailed Description	3449
4.926.2 Member Function Documentation	3449
4.926.3 Member Data Documentation	3451
4.927 <code>std::shared_future< _Res ></code> Class Template Reference	3452
4.927.1 Detailed Description	3453
4.927.2 Member Typedef Documentation	3453
4.927.3 Constructor & Destructor Documentation	3454
4.927.4 Member Function Documentation	3454
4.928 <code>std::shared_future< _Res & ></code> Class Template Reference	3455
4.928.1 Detailed Description	3456
4.928.2 Member Typedef Documentation	3457
4.928.3 Constructor & Destructor Documentation	3457
4.928.4 Member Function Documentation	3458
4.929 <code>std::shared_future< void ></code> Class Template Reference	3458
4.929.1 Detailed Description	3459
4.929.2 Member Typedef Documentation	3460
4.929.3 Constructor & Destructor Documentation	3460
4.929.4 Member Function Documentation	3461
4.930 <code>std::shared_lock< _Mutex ></code> Class Template Reference	3461
4.930.1 Detailed Description	3462
4.931 <code>std::shared_ptr< _Tp ></code> Class Template Reference	3462
4.931.1 Detailed Description	3465
4.931.2 Member Typedef Documentation	3466

4.931.3 Constructor & Destructor Documentation	3466
4.931.4 Member Function Documentation	3473
4.932 std::shared_timed_mutex Class Reference	3475
4.932.1 Detailed Description	3475
4.933 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference	3476
4.933.1 Detailed Description	3477
4.933.2 Member Typedef Documentation	3477
4.933.3 Constructor & Destructor Documentation	3477
4.933.4 Member Function Documentation	3479
4.933.5 Friends And Related Function Documentation	3481
4.934 std::slice Class Reference	3482
4.934.1 Detailed Description	3483
4.935 std::slice_array< _Tp > Class Template Reference	3483
4.935.1 Detailed Description	3484
4.936 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference	3484
4.936.1 Detailed Description	3487
4.937 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference	3487
4.937.1 Detailed Description	3490
4.937.2 Member Function Documentation	3490
4.938 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference	3491
4.938.1 Detailed Description	3492
4.939 __gnu_pbds::splay_tree_tag Struct Reference	3492
4.939.1 Detailed Description	3493
4.940 std::stack< _Tp, _Sequence > Class Template Reference	3493
4.940.1 Detailed Description	3494
4.940.2 Constructor & Destructor Documentation	3494
4.940.3 Member Function Documentation	3495
4.941 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference	3496
4.941.1 Detailed Description	3499
4.941.2 Constructor & Destructor Documentation	3499
4.941.3 Member Function Documentation	3501
4.941.4 Member Data Documentation	3520
4.942 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference	3524
4.942.1 Detailed Description	3527
4.942.2 Member Function Documentation	3527
4.942.3 Member Data Documentation	3544
4.943 std::chrono::_V2::steady_clock Struct Reference	3546
4.943.1 Detailed Description	3546

4.944	__gnu_pbds::detail::stored_data<_Tv,_Th,Store_Hash> Struct Template Reference	3547
4.944.1	Detailed Description	3547
4.945	__gnu_pbds::detail::stored_data<_Tv,_Th,false> Struct Template Reference	3548
4.945.1	Detailed Description	3548
4.946	__gnu_pbds::detail::stored_hash<_Th> Struct Template Reference	3549
4.946.1	Detailed Description	3549
4.947	__gnu_pbds::detail::stored_value<_Tv> Struct Template Reference	3550
4.947.1	Detailed Description	3550
4.948	__gnu_pbds::string_tag Struct Reference	3551
4.948.1	Detailed Description	3551
4.949	std::student_t_distribution<_RealType> Class Template Reference	3551
4.949.1	Detailed Description	3552
4.949.2	Member Typedef Documentation	3553
4.949.3	Member Function Documentation	3553
4.949.4	Friends And Related Function Documentation	3555
4.950	std::sub_match<_Bilter> Class Template Reference	3556
4.950.1	Detailed Description	3560
4.950.2	Member Typedef Documentation	3560
4.950.3	Member Function Documentation	3560
4.950.4	Friends And Related Function Documentation	3563
4.950.5	Member Data Documentation	3566
4.951	std::subtract_with_carry_engine<_UIntType, __w, __s, __r> Class Template Reference	3566
4.951.1	Detailed Description	3567
4.951.2	Member Typedef Documentation	3567
4.951.3	Constructor & Destructor Documentation	3568
4.951.4	Member Function Documentation	3568
4.951.5	Friends And Related Function Documentation	3570
4.952	__gnu_cxx::subtractive_rng Class Reference	3572
4.952.1	Detailed Description	3572
4.952.2	Member Typedef Documentation	3573
4.952.3	Constructor & Destructor Documentation	3573
4.952.4	Member Function Documentation	3573
4.953	__gnu_pbds::detail::synth_access_traits<Type_Traits,Set,_ATraits> Struct Template Reference	3574
4.953.1	Detailed Description	3574
4.954	std::chrono::_V2::system_clock Struct Reference	3575
4.954.1	Detailed Description	3575
4.955	std::system_error Class Reference	3576
4.955.1	Detailed Description	3576
4.955.2	Member Function Documentation	3577

4.956	__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp> Struct Template Reference	3577
4.956.1	Detailed Description	3578
4.956.2	Constructor & Destructor Documentation	3578
4.956.3	Member Function Documentation	3579
4.957	__gnu_pbds::detail::thin_heap<Value_Type, Cmp_Fn, _Alloc> Class Template Reference	3580
4.957.1	Detailed Description	3582
4.958	__gnu_pbds::thin_heap_tag Struct Reference	3582
4.958.1	Detailed Description	3583
4.959	std::thread Class Reference	3583
4.959.1	Detailed Description	3584
4.959.2	Member Function Documentation	3584
4.960	__gnu_cxx::throw_allocator_base<_Tp, _Cond> Class Template Reference	3584
4.960.1	Detailed Description	3585
4.961	__gnu_cxx::throw_allocator_limit<_Tp> Struct Template Reference	3586
4.961.1	Detailed Description	3587
4.962	__gnu_cxx::throw_allocator_random<_Tp> Struct Template Reference	3588
4.962.1	Detailed Description	3589
4.963	__gnu_cxx::throw_value_base<_Cond> Struct Template Reference	3589
4.963.1	Detailed Description	3590
4.964	__gnu_cxx::throw_value_limit Struct Reference	3591
4.964.1	Detailed Description	3592
4.965	__gnu_cxx::throw_value_random Struct Reference	3592
4.965.1	Detailed Description	3593
4.966	std::time_base Class Reference	3593
4.966.1	Detailed Description	3594
4.967	std::time_get<_CharT, _InIter> Class Template Reference	3594
4.967.1	Detailed Description	3596
4.967.2	Member Typedef Documentation	3596
4.967.3	Constructor & Destructor Documentation	3596
4.967.4	Member Function Documentation	3597
4.967.5	Member Data Documentation	3608
4.968	std::time_get_byname<_CharT, _InIter> Class Template Reference	3609
4.968.1	Detailed Description	3610
4.968.2	Member Function Documentation	3611
4.968.3	Member Data Documentation	3621
4.969	std::chrono::time_point<_Clock, _Dur> Struct Template Reference	3622
4.969.1	Detailed Description	3623
4.970	std::time_put<_CharT, _OutIter> Class Template Reference	3623
4.970.1	Detailed Description	3624

4.970.2 Member Typedef Documentation	3624
4.970.3 Constructor & Destructor Documentation	3625
4.970.4 Member Function Documentation	3626
4.970.5 Member Data Documentation	3628
4.971 std::time_put_byname< _CharT, _OutIter > Class Template Reference	3629
4.971.1 Detailed Description	3630
4.971.2 Member Function Documentation	3630
4.971.3 Member Data Documentation	3632
4.972 std::timed_mutex Class Reference	3633
4.972.1 Detailed Description	3633
4.973 std::to_chars_result Struct Reference	3633
4.973.1 Detailed Description	3633
4.974 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference	3634
4.974.1 Detailed Description	3634
4.975 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference	3634
4.975.1 Detailed Description	3635
4.975.2 Member Typedef Documentation	3635
4.975.3 Constructor & Destructor Documentation	3636
4.976 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference	3637
4.976.1 Detailed Description	3637
4.977 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference	3637
4.977.1 Detailed Description	3638
4.978 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference	3638
4.978.1 Detailed Description	3638
4.979 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference	3638
4.979.1 Detailed Description	3639
4.980 __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference	3639
4.980.1 Detailed Description	3640
4.980.2 Member Function Documentation	3640
4.981 __gnu_pbds::tree_tag Struct Reference	3642
4.981.1 Detailed Description	3642
4.982 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference	3642
4.982.1 Detailed Description	3642
4.983 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference	3643
4.983.1 Detailed Description	3643
4.983.2 Member Typedef Documentation	3643

4.984	__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	3644
4.984.1	Detailed Description	3644
4.984.2	Member Typedef Documentation	3645
4.985	__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	3645
4.985.1	Detailed Description	3646
4.985.2	Member Typedef Documentation	3646
4.986	__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference	3647
4.986.1	Detailed Description	3647
4.986.2	Member Typedef Documentation	3647
4.987	__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference	3648
4.987.1	Detailed Description	3648
4.987.2	Member Typedef Documentation	3649
4.988	__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference	3649
4.988.1	Detailed Description	3650
4.988.2	Member Typedef Documentation	3650
4.989	__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference	3651
4.989.1	Detailed Description	3651
4.989.2	Member Typedef Documentation	3652
4.989.3	Constructor & Destructor Documentation	3652
4.990	__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference	3653
4.990.1	Detailed Description	3653
4.991	__gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference	3654
4.991.1	Detailed Description	3654
4.992	__gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference	3654
4.992.1	Detailed Description	3654
4.993	__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference	3654
4.993.1	Detailed Description	3655
4.994	__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	3655
4.994.1	Detailed Description	3657
4.994.2	Member Function Documentation	3657
4.995	__gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	3658
4.995.1	Detailed Description	3660
4.996	__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc > Class Template Reference	3660

4.996.1 Detailed Description	3662
4.996.2 Member Typedef Documentation	3662
4.996.3 Member Function Documentation	3663
4.997 <code>__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc ></code> Struct Template Reference	3664
4.997.1 Detailed Description	3665
4.997.2 Member Typedef Documentation	3665
4.997.3 Member Function Documentation	3666
4.998 <code>__gnu_pbds::trie_tag</code> Struct Reference	3667
4.998.1 Detailed Description	3668
4.999 <code>__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc ></code> Struct Template Reference	3668
4.999.1 Detailed Description	3668
4.1000 <code>__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	3668
4.1000.1 Detailed Description	3669
4.1000.2 Member Typedef Documentation	3669
4.1001 <code>__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc ></code> Struct Template Reference	3670
4.1001.1 Detailed Description	3670
4.1001.2 Member Typedef Documentation	3670
4.1002 <code>__gnu_pbds::trivial_iterator_tag</code> Struct Reference	3671
4.1002.1 Detailed Description	3671
4.1003 <code>std::try_to_lock_t</code> Struct Reference	3671
4.1003.1 Detailed Description	3671
4.1004 <code>std::tuple< _Elements ></code> Class Template Reference	3672
4.1004.1 Detailed Description	3673
4.1005 <code>std::tuple< _T1, _T2 ></code> Class Template Reference	3674
4.1005.1 Detailed Description	3676
4.1006 <code>std::tuple_element< _Int, _Tp ></code> Struct Template Reference	3676
4.1006.1 Detailed Description	3676
4.1007 <code>std::tuple_element< 0, std::pair< _Tp1, _Tp2 > ></code> Struct Template Reference	3676
4.1007.1 Detailed Description	3676
4.1008 <code>std::tuple_element< 0, tuple< _Head, _Tail... > ></code> Struct Template Reference	3677
4.1008.1 Detailed Description	3677
4.1009 <code>std::tuple_element< 1, std::pair< _Tp1, _Tp2 > ></code> Struct Template Reference	3677
4.1009.1 Detailed Description	3677
4.1010 <code>std::tuple_element< __i, tuple< _Head, _Tail... > ></code> Struct Template Reference	3678
4.1010.1 Detailed Description	3678
4.1011 <code>std::tuple_element< __i, tuple< > ></code> Struct Template Reference	3678

4.1011.1 Detailed Description	3678
4.1012 std::tuple_element< _Int, ::array< _Tp, _Nm > > Struct Template Reference	3679
4.1012.1 Detailed Description	3679
4.1013 std::tuple_element< _Int, std::__debug::array< _Tp, _Nm > > Struct Template Reference	3679
4.1013.1 Detailed Description	3679
4.1014 std::tuple_size< _Tp > Struct Template Reference	3679
4.1014.1 Detailed Description	3680
4.1015 std::tuple_size< std::__debug::array< _Tp, _Nm > > Struct Template Reference	3680
4.1015.1 Detailed Description	3681
4.1016 std::tuple_size< std::pair< _Tp1, _Tp2 > > Struct Template Reference	3681
4.1016.1 Detailed Description	3682
4.1017 std::tuple_size< tuple< _Elements... > > Struct Template Reference	3682
4.1017.1 Detailed Description	3683
4.1018 std::tuple_size< ::array< _Tp, _Nm > > Struct Template Reference	3683
4.1018.1 Detailed Description	3684
4.1019 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference	3684
4.1019.1 Detailed Description	3684
4.1020 std::type_index Struct Reference	3685
4.1020.1 Detailed Description	3685
4.1021 std::type_info Class Reference	3685
4.1021.1 Detailed Description	3686
4.1021.2 Constructor & Destructor Documentation	3686
4.1021.3 Member Function Documentation	3686
4.1022 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference	3687
4.1022.1 Detailed Description	3688
4.1023 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference	3688
4.1023.1 Detailed Description	3689
4.1023.2 Member Typedef Documentation	3689
4.1024 std::unary_function< _Arg, _Result > Struct Template Reference	3689
4.1024.1 Detailed Description	3690
4.1024.2 Member Typedef Documentation	3690
4.1025 std::unary_negate< _Predicate > Class Template Reference	3691
4.1025.1 Detailed Description	3691
4.1025.2 Member Typedef Documentation	3691
4.1026 __gnu_parallel::unbalanced_tag Struct Reference	3692
4.1026.1 Detailed Description	3693
4.1026.2 Member Function Documentation	3693
4.1027 std::underflow_error Class Reference	3694
4.1027.1 Detailed Description	3694

4.1027.2 Member Function Documentation	3694
4.1028 std::underlying_type< _Tp > Struct Template Reference	3695
4.1028.1 Detailed Description	3695
4.1029 std::uniform_int_distribution< _IntType > Class Template Reference	3695
4.1029.1 Detailed Description	3696
4.1029.2 Member Typedef Documentation	3696
4.1029.3 Constructor & Destructor Documentation	3697
4.1029.4 Member Function Documentation	3697
4.1029.5 Friends And Related Function Documentation	3699
4.1030 std::uniform_real_distribution< _RealType > Class Template Reference	3699
4.1030.1 Detailed Description	3700
4.1030.2 Member Typedef Documentation	3700
4.1030.3 Constructor & Destructor Documentation	3700
4.1030.4 Member Function Documentation	3701
4.1030.5 Friends And Related Function Documentation	3702
4.1031 std::unique_lock< _Mutex > Class Template Reference	3703
4.1031.1 Detailed Description	3704
4.1031.2 Friends And Related Function Documentation	3704
4.1032 std::unique_ptr< _Tp, _Dp > Class Template Reference	3704
4.1032.1 Detailed Description	3706
4.1032.2 Constructor & Destructor Documentation	3707
4.1032.3 Member Function Documentation	3709
4.1033 std::unique_ptr< _Tp[], _Dp > Class Template Reference	3713
4.1033.1 Detailed Description	3714
4.1033.2 Constructor & Destructor Documentation	3714
4.1033.3 Member Function Documentation	3716
4.1034 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3719
4.1034.1 Detailed Description	3722
4.1034.2 Member Typedef Documentation	3723
4.1034.3 Constructor & Destructor Documentation	3727
4.1034.4 Member Function Documentation	3730
4.1035 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3753
4.1035.1 Detailed Description	3755
4.1035.2 Member Function Documentation	3755
4.1035.3 Member Data Documentation	3757
4.1036 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference	3759
4.1036.1 Detailed Description	3761
4.1036.2 Member Typedef Documentation	3762
4.1036.3 Constructor & Destructor Documentation	3766

4.1036.4 Member Function Documentation	3769
4.1037 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference .	3788
4.1037.1 Detailed Description	3791
4.1037.2 Member Function Documentation	3791
4.1037.3 Member Data Documentation	3793
4.1038 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3795
4.1038.1 Detailed Description	3797
4.1038.2 Member Typedef Documentation	3798
4.1038.3 Constructor & Destructor Documentation	3802
4.1038.4 Member Function Documentation	3804
4.1039 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . .	3824
4.1039.1 Detailed Description	3827
4.1039.2 Member Function Documentation	3827
4.1039.3 Member Data Documentation	3829
4.1040 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3830
4.1040.1 Detailed Description	3833
4.1040.2 Member Typedef Documentation	3833
4.1040.3 Constructor & Destructor Documentation	3837
4.1040.4 Member Function Documentation	3840
4.1041 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference	3859
4.1041.1 Detailed Description	3862
4.1041.2 Member Function Documentation	3862
4.1041.3 Member Data Documentation	3864
4.1042 std::uses_allocator< _Tp, _Alloc > Struct Template Reference	3865
4.1042.1 Detailed Description	3865
4.1043 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference	3866
4.1043.1 Detailed Description	3866
4.1044 std::valarray< _Tp > Class Template Reference	3867
4.1044.1 Detailed Description	3869
4.1044.2 Constructor & Destructor Documentation	3869
4.1045 std::vector< _Tp, _Alloc > Class Template Reference	3869
4.1045.1 Detailed Description	3873
4.1045.2 Constructor & Destructor Documentation	3873
4.1045.3 Member Function Documentation	3877
4.1046 std::__debug::vector< _Tp, _Allocator > Class Template Reference	3897
4.1046.1 Detailed Description	3899
4.1046.2 Constructor & Destructor Documentation	3899
4.1046.3 Member Function Documentation	3900
4.1046.4 Member Data Documentation	3901

4.1047 std::vector< bool, _Alloc > Class Template Reference	3902
4.1047.1 Detailed Description	3905
4.1048 std::wbuffer_convert< _Codecv, _Elem, _Tr > Class Template Reference	3906
4.1048.1 Detailed Description	3908
4.1048.2 Member Typedef Documentation	3908
4.1048.3 Constructor & Destructor Documentation	3910
4.1048.4 Member Function Documentation	3910
4.1048.5 Member Data Documentation	3925
4.1049 std::weak_ptr< _Tp > Class Template Reference	3927
4.1049.1 Detailed Description	3928
4.1050 std::weibull_distribution< _RealType > Class Template Reference	3928
4.1050.1 Detailed Description	3929
4.1050.2 Member Typedef Documentation	3929
4.1050.3 Member Function Documentation	3930
4.1050.4 Friends And Related Function Documentation	3932
4.1051 std::wstring_convert< _Codecv, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference	3932
4.1051.1 Detailed Description	3933
4.1051.2 Constructor & Destructor Documentation	3933
4.1051.3 Member Function Documentation	3936
5 File Documentation	3939
5.1 algo.h File Reference	3939
5.1.1 Detailed Description	3948
5.2 algbase.h File Reference	3948
5.2.1 Detailed Description	3950
5.3 algorithm File Reference	3950
5.3.1 Detailed Description	3950
5.4 algorithm File Reference	3950
5.4.1 Detailed Description	3952
5.5 algorithm File Reference	3952
5.5.1 Detailed Description	3952
5.6 algorithm File Reference	3952
5.6.1 Detailed Description	3953
5.6.2 Function Documentation	3953
5.7 algorithmfwd.h File Reference	3953
5.7.1 Detailed Description	3958
5.8 algorithmfwd.h File Reference	3959
5.8.1 Detailed Description	3967
5.9 aligned_buffer.h File Reference	3967

5.9.1 Detailed Description	3967
5.10 alloc_traits.h File Reference	3967
5.10.1 Detailed Description	3968
5.11 alloc_traits.h File Reference	3968
5.11.1 Detailed Description	3968
5.12 allocated_ptr.h File Reference	3969
5.12.1 Detailed Description	3969
5.13 allocator.h File Reference	3969
5.13.1 Detailed Description	3970
5.14 any File Reference	3970
5.14.1 Detailed Description	3970
5.15 any File Reference	3970
5.15.1 Detailed Description	3971
5.16 array File Reference	3971
5.16.1 Detailed Description	3972
5.17 array File Reference	3972
5.17.1 Detailed Description	3973
5.18 array File Reference	3973
5.18.1 Detailed Description	3973
5.19 assertions.h File Reference	3973
5.19.1 Detailed Description	3974
5.20 assoc_container.hpp File Reference	3974
5.20.1 Detailed Description	3974
5.21 atomic File Reference	3975
5.21.1 Detailed Description	3979
5.22 atomic_base.h File Reference	3979
5.22.1 Detailed Description	3980
5.23 atomic_futex.h File Reference	3980
5.23.1 Detailed Description	3980
5.24 atomic_lockfree_defines.h File Reference	3980
5.24.1 Detailed Description	3980
5.25 atomic_word.h File Reference	3980
5.25.1 Detailed Description	3981
5.26 atomicity.h File Reference	3981
5.26.1 Detailed Description	3981
5.27 auto_ptr.h File Reference	3981
5.27.1 Detailed Description	3982
5.28 backward_warning.h File Reference	3982
5.28.1 Detailed Description	3982

5.29	balanced_quicksort.h File Reference	3982
5.29.1	Detailed Description	3982
5.30	base.h File Reference	3983
5.30.1	Detailed Description	3983
5.31	basic_file.h File Reference	3984
5.31.1	Detailed Description	3984
5.32	basic_ios.h File Reference	3984
5.32.1	Detailed Description	3984
5.33	basic_ios.tcc File Reference	3984
5.33.1	Detailed Description	3985
5.34	basic_iterator.h File Reference	3985
5.34.1	Detailed Description	3985
5.35	basic_string.h File Reference	3985
5.35.1	Detailed Description	3988
5.36	basic_string.tcc File Reference	3988
5.36.1	Detailed Description	3988
5.37	bin_search_tree_.hpp File Reference	3988
5.37.1	Detailed Description	3989
5.38	binary_heap_.hpp File Reference	3989
5.38.1	Detailed Description	3989
5.39	binders.h File Reference	3989
5.39.1	Detailed Description	3990
5.40	binomial_heap_.hpp File Reference	3990
5.40.1	Detailed Description	3990
5.41	binomial_heap_base_.hpp File Reference	3990
5.41.1	Detailed Description	3991
5.42	bit File Reference	3991
5.42.1	Detailed Description	3991
5.43	bitmap_allocator.h File Reference	3991
5.43.1	Detailed Description	3992
5.43.2	Macro Definition Documentation	3992
5.44	bitset File Reference	3993
5.44.1	Detailed Description	3993
5.45	bitset File Reference	3994
5.45.1	Detailed Description	3994
5.46	bool_set File Reference	3994
5.46.1	Detailed Description	3995
5.47	bool_set.tcc File Reference	3995
5.47.1	Detailed Description	3995

5.48 boost_concept_check.h File Reference	3996
5.48.1 Detailed Description	3996
5.49 branch_policy.hpp File Reference	3996
5.49.1 Detailed Description	3996
5.50 c++0x_warning.h File Reference	3997
5.50.1 Detailed Description	3997
5.51 c++allocator.h File Reference	3997
5.51.1 Detailed Description	3997
5.52 c++config.h File Reference	3997
5.52.1 Detailed Description	4003
5.53 c++io.h File Reference	4003
5.53.1 Detailed Description	4003
5.54 c++locale.h File Reference	4003
5.54.1 Detailed Description	4004
5.55 c++locale_internal.h File Reference	4004
5.55.1 Detailed Description	4004
5.56 cassert File Reference	4004
5.56.1 Detailed Description	4004
5.57 cast.h File Reference	4004
5.57.1 Detailed Description	4005
5.58 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference	4005
5.58.1 Detailed Description	4005
5.59 cc_ht_map_.hpp File Reference	4005
5.59.1 Detailed Description	4006
5.60 ccomplex File Reference	4006
5.60.1 Detailed Description	4006
5.61 ccomplex File Reference	4006
5.61.1 Detailed Description	4006
5.62 cctype File Reference	4006
5.62.1 Detailed Description	4007
5.63 cctype File Reference	4007
5.63.1 Detailed Description	4007
5.64 cerrno File Reference	4007
5.64.1 Detailed Description	4007
5.65 cfenv File Reference	4007
5.65.1 Detailed Description	4008
5.66 cfenv File Reference	4008
5.66.1 Detailed Description	4008
5.67 cfloat File Reference	4008

5.67.1 Detailed Description	4008
5.68 cfloat File Reference	4008
5.68.1 Detailed Description	4008
5.69 char_traits.h File Reference	4009
5.69.1 Detailed Description	4009
5.70 charconv File Reference	4009
5.70.1 Detailed Description	4011
5.71 charconv.h File Reference	4011
5.71.1 Detailed Description	4011
5.72 checkers.h File Reference	4011
5.72.1 Detailed Description	4012
5.73 chrono File Reference	4012
5.73.1 Detailed Description	4013
5.74 chrono File Reference	4013
5.74.1 Detailed Description	4014
5.75 cinttypes File Reference	4014
5.75.1 Detailed Description	4014
5.76 cinttypes File Reference	4014
5.76.1 Detailed Description	4014
5.77 ciso646 File Reference	4014
5.77.1 Detailed Description	4014
5.78 climits File Reference	4015
5.78.1 Detailed Description	4015
5.79 climits File Reference	4015
5.79.1 Detailed Description	4015
5.80 clocale File Reference	4015
5.80.1 Detailed Description	4015
5.81 cmath File Reference	4016
5.81.1 Detailed Description	4018
5.82 cmath File Reference	4018
5.82.1 Detailed Description	4018
5.83 cmath File Reference	4018
5.83.1 Detailed Description	4021
5.84 cmp_fn_imps.hpp File Reference	4021
5.84.1 Detailed Description	4021
5.85 codecvt File Reference	4021
5.85.1 Detailed Description	4021
5.86 codecvt.h File Reference	4021
5.86.1 Detailed Description	4022

5.87 codecvt_specializations.h File Reference	4022
5.87.1 Detailed Description	4022
5.88 compare File Reference	4022
5.88.1 Detailed Description	4022
5.89 compatibility.h File Reference	4022
5.89.1 Detailed Description	4022
5.90 compatibility.h File Reference	4023
5.90.1 Detailed Description	4023
5.91 compiletime_settings.h File Reference	4023
5.91.1 Detailed Description	4023
5.91.2 Macro Definition Documentation	4023
5.92 complex File Reference	4025
5.92.1 Detailed Description	4029
5.93 complex File Reference	4030
5.93.1 Detailed Description	4030
5.94 complex.h File Reference	4030
5.94.1 Detailed Description	4031
5.95 concept_check.h File Reference	4031
5.95.1 Detailed Description	4031
5.96 concepts File Reference	4031
5.96.1 Detailed Description	4031
5.97 concurrency.h File Reference	4031
5.97.1 Detailed Description	4032
5.98 cond_dealtor.hpp File Reference	4032
5.98.1 Detailed Description	4032
5.99 cond_key_dtor_entry_dealtor.hpp File Reference	4032
5.99.1 Detailed Description	4032
5.100 condition_variable File Reference	4033
5.100.1 Detailed Description	4033
5.101 const_iterator.hpp File Reference	4033
5.101.1 Detailed Description	4034
5.102 const_iterator.hpp File Reference	4034
5.102.1 Detailed Description	4034
5.103 const_iterator.hpp File Reference	4034
5.103.1 Detailed Description	4034
5.104 constructor_destructor_fn_imps.hpp File Reference	4034
5.104.1 Detailed Description	4034
5.105 constructor_destructor_fn_imps.hpp File Reference	4035
5.105.1 Detailed Description	4035

5.106 constructor_destructor_fn_imps.hpp File Reference	4035
5.107 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	4035
5.107.1 Detailed Description	4035
5.108 constructor_destructor_no_store_hash_fn_imps.hpp File Reference	4035
5.108.1 Detailed Description	4035
5.109 constructor_destructor_store_hash_fn_imps.hpp File Reference	4035
5.109.1 Detailed Description	4035
5.110 constructor_destructor_store_hash_fn_imps.hpp File Reference	4035
5.110.1 Detailed Description	4035
5.111 constructors_destructor_fn_imps.hpp File Reference	4035
5.111.1 Detailed Description	4035
5.112 constructors_destructor_fn_imps.hpp File Reference	4036
5.112.1 Detailed Description	4036
5.113 constructors_destructor_fn_imps.hpp File Reference	4036
5.113.1 Detailed Description	4036
5.114 constructors_destructor_fn_imps.hpp File Reference	4036
5.114.1 Detailed Description	4036
5.115 constructors_destructor_fn_imps.hpp File Reference	4036
5.115.1 Detailed Description	4036
5.116 constructors_destructor_fn_imps.hpp File Reference	4036
5.116.1 Detailed Description	4036
5.117 constructors_destructor_fn_imps.hpp File Reference	4036
5.117.1 Detailed Description	4036
5.118 constructors_destructor_fn_imps.hpp File Reference	4037
5.118.1 Detailed Description	4037
5.119 constructors_destructor_fn_imps.hpp File Reference	4037
5.119.1 Detailed Description	4037
5.120 constructors_destructor_fn_imps.hpp File Reference	4037
5.120.1 Detailed Description	4037
5.121 constructors_destructor_fn_imps.hpp File Reference	4037
5.121.1 Detailed Description	4037
5.122 constructors_destructor_fn_imps.hpp File Reference	4037
5.122.1 Detailed Description	4037
5.123 container_base_dispatch.hpp File Reference	4037
5.123.1 Detailed Description	4038
5.124 cpp_type_traits.h File Reference	4038
5.124.1 Detailed Description	4039
5.125 cpu_defines.h File Reference	4039
5.125.1 Detailed Description	4039

5.126 csetjmp File Reference	4039
5.126.1 Detailed Description	4039
5.127 csignal File Reference	4039
5.127.1 Detailed Description	4040
5.128 cstdalign File Reference	4040
5.128.1 Detailed Description	4040
5.129 cstdarg File Reference	4040
5.129.1 Detailed Description	4040
5.130 cstdarg File Reference	4040
5.130.1 Detailed Description	4041
5.131 cstdlib File Reference	4041
5.131.1 Detailed Description	4041
5.132 cstdlib File Reference	4041
5.132.1 Detailed Description	4041
5.133 cstdint File Reference	4041
5.133.1 Detailed Description	4041
5.134 cstdint File Reference	4042
5.134.1 Detailed Description	4042
5.135 cstdint File Reference	4042
5.135.1 Detailed Description	4042
5.136 cstdio File Reference	4042
5.136.1 Detailed Description	4043
5.137 cstdio File Reference	4043
5.137.1 Detailed Description	4043
5.138 cstdlib File Reference	4043
5.138.1 Detailed Description	4043
5.139 cstdlib File Reference	4044
5.139.1 Detailed Description	4044
5.140 cstring File Reference	4044
5.140.1 Detailed Description	4044
5.141 ctgmath File Reference	4044
5.141.1 Detailed Description	4045
5.142 ctgmath File Reference	4045
5.142.1 Detailed Description	4045
5.143 ctime File Reference	4045
5.143.1 Detailed Description	4045
5.144 ctime File Reference	4045
5.144.1 Detailed Description	4045
5.145 ctype_base.h File Reference	4046

5.145.1 Detailed Description	4046
5.146 ctype_inline.h File Reference	4046
5.146.1 Detailed Description	4046
5.147 cuchar File Reference	4046
5.147.1 Detailed Description	4046
5.148 wchar File Reference	4047
5.148.1 Detailed Description	4047
5.149 wchar File Reference	4047
5.149.1 Detailed Description	4047
5.150 wctype File Reference	4048
5.150.1 Detailed Description	4048
5.151 wctype File Reference	4048
5.151.1 Detailed Description	4048
5.152 cxxabi.h File Reference	4048
5.152.1 Detailed Description	4050
5.152.2 Function Documentation	4050
5.153 cxxabi_forced.h File Reference	4051
5.153.1 Detailed Description	4051
5.154 cxxabi_init_exception.h File Reference	4051
5.154.1 Detailed Description	4051
5.155 cxxabi_tweaks.h File Reference	4051
5.155.1 Detailed Description	4052
5.156 debug.h File Reference	4052
5.156.1 Detailed Description	4053
5.157 debug_allocator.h File Reference	4053
5.157.1 Detailed Description	4053
5.158 debug_fn_imps.hpp File Reference	4053
5.158.1 Detailed Description	4053
5.159 debug_fn_imps.hpp File Reference	4053
5.159.1 Detailed Description	4053
5.160 debug_fn_imps.hpp File Reference	4053
5.160.1 Detailed Description	4053
5.161 debug_fn_imps.hpp File Reference	4053
5.161.1 Detailed Description	4053
5.162 debug_fn_imps.hpp File Reference	4054
5.162.1 Detailed Description	4054
5.163 debug_fn_imps.hpp File Reference	4054
5.163.1 Detailed Description	4054
5.164 debug_fn_imps.hpp File Reference	4054

5.164.1 Detailed Description	4054
5.165 debug_fn_imps.hpp File Reference	4054
5.165.1 Detailed Description	4054
5.166 debug_fn_imps.hpp File Reference	4054
5.166.1 Detailed Description	4054
5.167 debug_fn_imps.hpp File Reference	4054
5.167.1 Detailed Description	4054
5.168 debug_fn_imps.hpp File Reference	4055
5.168.1 Detailed Description	4055
5.169 debug_fn_imps.hpp File Reference	4055
5.169.1 Detailed Description	4055
5.170 debug_fn_imps.hpp File Reference	4055
5.170.1 Detailed Description	4055
5.171 debug_fn_imps.hpp File Reference	4055
5.171.1 Detailed Description	4055
5.172 debug_fn_imps.hpp File Reference	4055
5.172.1 Detailed Description	4055
5.173 debug_map_base.hpp File Reference	4055
5.173.1 Detailed Description	4055
5.174 debug_no_store_hash_fn_imps.hpp File Reference	4056
5.174.1 Detailed Description	4056
5.175 debug_no_store_hash_fn_imps.hpp File Reference	4056
5.175.1 Detailed Description	4056
5.176 debug_store_hash_fn_imps.hpp File Reference	4056
5.176.1 Detailed Description	4056
5.177 debug_store_hash_fn_imps.hpp File Reference	4056
5.177.1 Detailed Description	4056
5.178 decimal File Reference	4056
5.178.1 Detailed Description	4065
5.179 deque File Reference	4066
5.179.1 Detailed Description	4066
5.180 deque File Reference	4066
5.180.1 Detailed Description	4067
5.181 deque File Reference	4067
5.181.1 Detailed Description	4067
5.182 deque.tcc File Reference	4067
5.182.1 Detailed Description	4068
5.183 direct_mask_range_hashing_imp.hpp File Reference	4069
5.183.1 Detailed Description	4069

5.184 direct_mod_range_hashing_imp.hpp File Reference	4069
5.184.1 Detailed Description	4069
5.185 dynamic_bitset File Reference	4069
5.185.1 Detailed Description	4070
5.186 dynamic_bitset.tcc File Reference	4070
5.186.1 Detailed Description	4071
5.187 enable_special_members.h File Reference	4071
5.187.1 Detailed Description	4071
5.188 enc_filebuf.h File Reference	4071
5.188.1 Detailed Description	4071
5.189 entry_cmp.hpp File Reference	4071
5.189.1 Detailed Description	4072
5.190 entry_list_fn_imps.hpp File Reference	4072
5.190.1 Detailed Description	4072
5.191 entry_metadata_base.hpp File Reference	4072
5.191.1 Detailed Description	4072
5.192 entry_pred.hpp File Reference	4072
5.192.1 Detailed Description	4072
5.193 eq_by_less.hpp File Reference	4073
5.193.1 Detailed Description	4073
5.194 equally_split.h File Reference	4073
5.194.1 Detailed Description	4073
5.195 erase_fn_imps.hpp File Reference	4073
5.195.1 Detailed Description	4073
5.196 erase_fn_imps.hpp File Reference	4074
5.196.1 Detailed Description	4074
5.197 erase_fn_imps.hpp File Reference	4074
5.197.1 Detailed Description	4074
5.198 erase_fn_imps.hpp File Reference	4074
5.198.1 Detailed Description	4074
5.199 erase_fn_imps.hpp File Reference	4074
5.199.1 Detailed Description	4074
5.200 erase_fn_imps.hpp File Reference	4074
5.200.1 Detailed Description	4074
5.201 erase_fn_imps.hpp File Reference	4074
5.201.1 Detailed Description	4074
5.202 erase_fn_imps.hpp File Reference	4075
5.202.1 Detailed Description	4075
5.203 erase_fn_imps.hpp File Reference	4075

5.203.1 Detailed Description	4075
5.204 erase_fn_imps.hpp File Reference	4075
5.204.1 Detailed Description	4075
5.205 erase_fn_imps.hpp File Reference	4075
5.205.1 Detailed Description	4075
5.206 erase_fn_imps.hpp File Reference	4075
5.206.1 Detailed Description	4075
5.207 erase_fn_imps.hpp File Reference	4075
5.207.1 Detailed Description	4075
5.208 erase_fn_imps.hpp File Reference	4076
5.208.1 Detailed Description	4076
5.209 erase_if.h File Reference	4076
5.209.1 Detailed Description	4076
5.210 erase_no_store_hash_fn_imps.hpp File Reference	4076
5.210.1 Detailed Description	4076
5.211 erase_no_store_hash_fn_imps.hpp File Reference	4076
5.211.1 Detailed Description	4076
5.212 erase_store_hash_fn_imps.hpp File Reference	4076
5.212.1 Detailed Description	4076
5.213 erase_store_hash_fn_imps.hpp File Reference	4077
5.213.1 Detailed Description	4077
5.214 error_constants.h File Reference	4077
5.214.1 Detailed Description	4077
5.215 exception File Reference	4077
5.215.1 Detailed Description	4078
5.216 exception.h File Reference	4078
5.216.1 Detailed Description	4078
5.217 exception.hpp File Reference	4079
5.217.1 Detailed Description	4079
5.218 exception_defines.h File Reference	4079
5.218.1 Detailed Description	4079
5.219 exception_ptr.h File Reference	4079
5.219.1 Detailed Description	4080
5.220 extc++.h File Reference	4080
5.220.1 Detailed Description	4080
5.221 extptr_allocator.h File Reference	4080
5.221.1 Detailed Description	4080
5.222 features.h File Reference	4081
5.222.1 Detailed Description	4081

5.222.2 Macro Definition Documentation	4081
5.223 fenv.h File Reference	4083
5.223.1 Detailed Description	4083
5.224 filesystem File Reference	4084
5.224.1 Detailed Description	4084
5.225 filesystem File Reference	4084
5.225.1 Detailed Description	4084
5.226 find.h File Reference	4084
5.226.1 Detailed Description	4085
5.227 find_fn_imps.hpp File Reference	4085
5.227.1 Detailed Description	4085
5.228 find_fn_imps.hpp File Reference	4085
5.228.1 Detailed Description	4085
5.229 find_fn_imps.hpp File Reference	4085
5.229.1 Detailed Description	4085
5.230 find_fn_imps.hpp File Reference	4085
5.230.1 Detailed Description	4085
5.231 find_fn_imps.hpp File Reference	4085
5.231.1 Detailed Description	4085
5.232 find_fn_imps.hpp File Reference	4085
5.232.1 Detailed Description	4085
5.233 find_fn_imps.hpp File Reference	4086
5.233.1 Detailed Description	4086
5.234 find_fn_imps.hpp File Reference	4086
5.234.1 Detailed Description	4086
5.235 find_fn_imps.hpp File Reference	4086
5.235.1 Detailed Description	4086
5.236 find_fn_imps.hpp File Reference	4086
5.236.1 Detailed Description	4086
5.237 find_fn_imps.hpp File Reference	4086
5.237.1 Detailed Description	4086
5.238 find_no_store_hash_fn_imps.hpp File Reference	4086
5.238.1 Detailed Description	4086
5.239 find_selectors.h File Reference	4087
5.239.1 Detailed Description	4087
5.240 find_store_hash_fn_imps.hpp File Reference	4087
5.240.1 Detailed Description	4087
5.241 find_store_hash_fn_imps.hpp File Reference	4087
5.241.1 Detailed Description	4087

5.242 for_each.h File Reference	4087
5.242.1 Detailed Description	4088
5.243 for_each_selectors.h File Reference	4088
5.243.1 Detailed Description	4088
5.244 formatter.h File Reference	4088
5.244.1 Detailed Description	4089
5.245 forward_list File Reference	4089
5.245.1 Detailed Description	4090
5.246 forward_list File Reference	4090
5.246.1 Detailed Description	4091
5.247 forward_list File Reference	4091
5.247.1 Detailed Description	4091
5.248 forward_list.h File Reference	4091
5.248.1 Detailed Description	4092
5.249 forward_list.tcc File Reference	4092
5.249.1 Detailed Description	4093
5.250 fs_dir.h File Reference	4093
5.250.1 Detailed Description	4093
5.251 fs_dir.h File Reference	4093
5.251.1 Detailed Description	4093
5.252 fs_fwd.h File Reference	4093
5.252.1 Detailed Description	4093
5.253 fs_fwd.h File Reference	4094
5.253.1 Detailed Description	4095
5.254 fs_path.h File Reference	4095
5.254.1 Detailed Description	4095
5.255 fs_path.h File Reference	4096
5.255.1 Detailed Description	4096
5.256 fstream File Reference	4096
5.256.1 Detailed Description	4097
5.257 fstream.tcc File Reference	4097
5.257.1 Detailed Description	4097
5.258 funtexcept.h File Reference	4097
5.258.1 Detailed Description	4098
5.259 functional File Reference	4098
5.259.1 Detailed Description	4100
5.260 functional File Reference	4100
5.260.1 Detailed Description	4101
5.261 functional File Reference	4101

5.261.1 Detailed Description	4102
5.261.2 Function Documentation	4102
5.261.3 Variable Documentation	4103
5.262 functional_hash.h File Reference	4103
5.262.1 Detailed Description	4104
5.263 functions.h File Reference	4104
5.263.1 Detailed Description	4106
5.264 future File Reference	4106
5.264.1 Detailed Description	4107
5.265 gp_ht_map_.hpp File Reference	4107
5.265.1 Detailed Description	4108
5.266 gslice.h File Reference	4108
5.266.1 Detailed Description	4108
5.267 gslice_array.h File Reference	4108
5.267.1 Detailed Description	4109
5.268 hash_bytes.h File Reference	4109
5.268.1 Detailed Description	4109
5.269 hash_eq_fn.hpp File Reference	4109
5.269.1 Detailed Description	4109
5.270 hash_exponential_size_policy_imp.hpp File Reference	4110
5.270.1 Detailed Description	4110
5.271 hash_fun.h File Reference	4110
5.271.1 Detailed Description	4110
5.272 hash_load_check_resize_trigger_imp.hpp File Reference	4110
5.272.1 Detailed Description	4110
5.273 hash_load_check_resize_trigger_size_base.hpp File Reference	4110
5.273.1 Detailed Description	4111
5.274 hash_map File Reference	4111
5.274.1 Detailed Description	4111
5.275 hash_policy.hpp File Reference	4112
5.275.1 Detailed Description	4113
5.276 hash_prime_size_policy_imp.hpp File Reference	4113
5.276.1 Detailed Description	4113
5.277 hash_set File Reference	4113
5.277.1 Detailed Description	4114
5.278 hash_standard_resize_policy_imp.hpp File Reference	4114
5.278.1 Detailed Description	4114
5.279 hashtable.h File Reference	4114
5.279.1 Detailed Description	4114

5.280 hashtable.h File Reference	4114
5.280.1 Detailed Description	4115
5.281 hashtable_policy.h File Reference	4115
5.281.1 Detailed Description	4117
5.282 helper_functions.h File Reference	4117
5.282.1 Detailed Description	4118
5.283 indirect_array.h File Reference	4118
5.283.1 Detailed Description	4119
5.284 info_fn_imps.hpp File Reference	4119
5.284.1 Detailed Description	4119
5.285 info_fn_imps.hpp File Reference	4119
5.285.1 Detailed Description	4119
5.286 info_fn_imps.hpp File Reference	4119
5.286.1 Detailed Description	4119
5.287 info_fn_imps.hpp File Reference	4119
5.287.1 Detailed Description	4119
5.288 info_fn_imps.hpp File Reference	4119
5.288.1 Detailed Description	4119
5.289 info_fn_imps.hpp File Reference	4119
5.289.1 Detailed Description	4119
5.290 info_fn_imps.hpp File Reference	4120
5.290.1 Detailed Description	4120
5.291 info_fn_imps.hpp File Reference	4120
5.291.1 Detailed Description	4120
5.292 info_fn_imps.hpp File Reference	4120
5.292.1 Detailed Description	4120
5.293 info_fn_imps.hpp File Reference	4120
5.293.1 Detailed Description	4120
5.294 initializer_list File Reference	4120
5.294.1 Detailed Description	4120
5.295 insert_fn_imps.hpp File Reference	4121
5.295.1 Detailed Description	4121
5.296 insert_fn_imps.hpp File Reference	4121
5.296.1 Detailed Description	4121
5.297 insert_fn_imps.hpp File Reference	4121
5.297.1 Detailed Description	4121
5.298 insert_fn_imps.hpp File Reference	4121
5.298.1 Detailed Description	4121
5.299 insert_fn_imps.hpp File Reference	4121

5.299.1 Detailed Description	4121
5.300 insert_fn_imps.hpp File Reference	4121
5.300.1 Detailed Description	4121
5.301 insert_fn_imps.hpp File Reference	4122
5.301.1 Detailed Description	4122
5.302 insert_fn_imps.hpp File Reference	4122
5.302.1 Detailed Description	4122
5.303 insert_fn_imps.hpp File Reference	4122
5.303.1 Detailed Description	4122
5.304 insert_fn_imps.hpp File Reference	4122
5.304.1 Detailed Description	4122
5.305 insert_fn_imps.hpp File Reference	4122
5.305.1 Detailed Description	4122
5.306 insert_fn_imps.hpp File Reference	4122
5.306.1 Detailed Description	4122
5.307 insert_fn_imps.hpp File Reference	4123
5.307.1 Detailed Description	4123
5.308 insert_join_fn_imps.hpp File Reference	4123
5.308.1 Detailed Description	4123
5.309 insert_no_store_hash_fn_imps.hpp File Reference	4123
5.309.1 Detailed Description	4123
5.310 insert_no_store_hash_fn_imps.hpp File Reference	4123
5.310.1 Detailed Description	4123
5.311 insert_store_hash_fn_imps.hpp File Reference	4123
5.311.1 Detailed Description	4123
5.312 insert_store_hash_fn_imps.hpp File Reference	4123
5.312.1 Detailed Description	4123
5.313 invoke.h File Reference	4123
5.313.1 Detailed Description	4124
5.314 iomanip File Reference	4124
5.314.1 Detailed Description	4126
5.315 ios File Reference	4126
5.315.1 Detailed Description	4126
5.316 ios_base.h File Reference	4126
5.316.1 Detailed Description	4128
5.317 iosfwd File Reference	4128
5.317.1 Detailed Description	4129
5.318 istream File Reference	4129
5.318.1 Detailed Description	4130

5.319 istream File Reference	4130
5.319.1 Detailed Description	4131
5.320 istream.tcc File Reference	4131
5.320.1 Detailed Description	4132
5.321 iterator File Reference	4132
5.321.1 Detailed Description	4132
5.322 iterator File Reference	4132
5.322.1 Detailed Description	4133
5.323 iterator File Reference	4133
5.323.1 Detailed Description	4133
5.323.2 Function Documentation	4133
5.324 iterator.h File Reference	4134
5.324.1 Detailed Description	4134
5.325 iterator.hpp File Reference	4134
5.325.1 Detailed Description	4134
5.326 iterator_concepts.h File Reference	4134
5.326.1 Detailed Description	4134
5.327 iterator_fn_imps.hpp File Reference	4135
5.327.1 Detailed Description	4135
5.328 iterators_fn_imps.hpp File Reference	4135
5.328.1 Detailed Description	4135
5.329 iterators_fn_imps.hpp File Reference	4135
5.329.1 Detailed Description	4135
5.330 iterators_fn_imps.hpp File Reference	4135
5.330.1 Detailed Description	4135
5.331 iterators_fn_imps.hpp File Reference	4135
5.331.1 Detailed Description	4135
5.332 iterators_fn_imps.hpp File Reference	4135
5.332.1 Detailed Description	4135
5.333 iterators_fn_imps.hpp File Reference	4136
5.333.1 Detailed Description	4136
5.334 iterators_fn_imps.hpp File Reference	4136
5.334.1 Detailed Description	4136
5.335 left_child_next_sibling_heap_.hpp File Reference	4136
5.335.1 Detailed Description	4136
5.336 lfts_config.h File Reference	4136
5.336.1 Detailed Description	4136
5.337 limits File Reference	4137
5.337.1 Detailed Description	4138

5.338 linear_probe_fn_imp.hpp File Reference	4138
5.338.1 Detailed Description	4138
5.339 list File Reference	4138
5.339.1 Detailed Description	4138
5.340 list File Reference	4138
5.340.1 Detailed Description	4139
5.341 list File Reference	4139
5.341.1 Detailed Description	4140
5.342 list.tcc File Reference	4140
5.342.1 Detailed Description	4140
5.343 list_partition.h File Reference	4140
5.343.1 Detailed Description	4141
5.344 list_update_policy.hpp File Reference	4141
5.344.1 Detailed Description	4141
5.345 locale File Reference	4141
5.345.1 Detailed Description	4141
5.346 locale_classes.h File Reference	4142
5.346.1 Detailed Description	4142
5.347 locale_classes.tcc File Reference	4142
5.347.1 Detailed Description	4142
5.348 locale_conv.h File Reference	4143
5.348.1 Detailed Description	4143
5.349 locale_facets.h File Reference	4143
5.349.1 Detailed Description	4145
5.350 locale_facets.tcc File Reference	4145
5.350.1 Detailed Description	4145
5.351 locale_facets_nonio.h File Reference	4146
5.351.1 Detailed Description	4146
5.352 locale_facets_nonio.tcc File Reference	4146
5.352.1 Detailed Description	4146
5.353 localefwd.h File Reference	4147
5.353.1 Detailed Description	4148
5.354 losertree.h File Reference	4148
5.354.1 Detailed Description	4148
5.355 lu_counter_metadata.hpp File Reference	4149
5.355.1 Detailed Description	4149
5.356 lu_map_.hpp File Reference	4149
5.356.1 Detailed Description	4149
5.357 macros.h File Reference	4150

5.357.1 Detailed Description	4150
5.357.2 Macro Definition Documentation	4151
5.358 malloc_allocator.h File Reference	4154
5.358.1 Detailed Description	4154
5.359 map File Reference	4154
5.359.1 Detailed Description	4155
5.360 map File Reference	4155
5.360.1 Detailed Description	4155
5.361 map File Reference	4155
5.361.1 Detailed Description	4156
5.362 map.h File Reference	4156
5.362.1 Detailed Description	4157
5.363 mask_array.h File Reference	4157
5.363.1 Detailed Description	4157
5.364 mask_based_range_hashing.hpp File Reference	4157
5.364.1 Detailed Description	4157
5.365 math.h File Reference	4157
5.365.1 Detailed Description	4157
5.366 memory File Reference	4158
5.366.1 Detailed Description	4158
5.367 memory File Reference	4158
5.367.1 Detailed Description	4159
5.368 memory File Reference	4159
5.368.1 Detailed Description	4160
5.369 memory_resource File Reference	4160
5.369.1 Detailed Description	4160
5.370 memory_resource File Reference	4160
5.370.1 Detailed Description	4161
5.370.2 Function Documentation	4161
5.371 memoryfwd.h File Reference	4162
5.371.1 Detailed Description	4162
5.372 merge.h File Reference	4162
5.372.1 Detailed Description	4163
5.373 messages_members.h File Reference	4163
5.373.1 Detailed Description	4163
5.374 mod_based_range_hashing.hpp File Reference	4163
5.374.1 Detailed Description	4163
5.375 move.h File Reference	4163
5.375.1 Detailed Description	4164

5.376	mt_allocator.h File Reference	4164
5.376.1	Detailed Description	4165
5.377	multimap.h File Reference	4165
5.377.1	Detailed Description	4166
5.378	multiseq_selection.h File Reference	4166
5.378.1	Detailed Description	4166
5.379	multiset.h File Reference	4167
5.379.1	Detailed Description	4167
5.380	multiway_merge.h File Reference	4167
5.380.1	Detailed Description	4170
5.380.2	Macro Definition Documentation	4170
5.381	multiway_mergesort.h File Reference	4170
5.381.1	Detailed Description	4171
5.382	mutex File Reference	4171
5.382.1	Detailed Description	4172
5.383	nested_exception.h File Reference	4172
5.383.1	Detailed Description	4172
5.384	net.h File Reference	4172
5.384.1	Detailed Description	4172
5.385	new File Reference	4172
5.385.1	Detailed Description	4173
5.385.2	Function Documentation	4173
5.386	new_allocator.h File Reference	4178
5.386.1	Detailed Description	4178
5.387	node.hpp File Reference	4178
5.387.1	Detailed Description	4178
5.388	node.hpp File Reference	4178
5.388.1	Detailed Description	4178
5.389	node.hpp File Reference	4179
5.389.1	Detailed Description	4179
5.390	node_handle.h File Reference	4179
5.390.1	Detailed Description	4179
5.391	node_iterators.hpp File Reference	4179
5.391.1	Detailed Description	4179
5.392	node_iterators.hpp File Reference	4180
5.392.1	Detailed Description	4180
5.393	node_metadata_selector.hpp File Reference	4180
5.393.1	Detailed Description	4180
5.394	node_metadata_selector.hpp File Reference	4180

5.394.1 Detailed Description	4181
5.395 null_node_metadata.hpp File Reference	4181
5.395.1 Detailed Description	4181
5.396 numbers File Reference	4181
5.396.1 Detailed Description	4181
5.397 numeric File Reference	4181
5.397.1 Detailed Description	4182
5.398 numeric File Reference	4182
5.398.1 Detailed Description	4182
5.399 numeric File Reference	4183
5.399.1 Detailed Description	4184
5.400 numeric File Reference	4185
5.400.1 Detailed Description	4185
5.400.2 Function Documentation	4185
5.401 numeric_traits.h File Reference	4186
5.401.1 Detailed Description	4186
5.402 numeric_fwd.h File Reference	4186
5.402.1 Detailed Description	4188
5.403 omp_loop.h File Reference	4188
5.403.1 Detailed Description	4188
5.404 omp_loop_static.h File Reference	4188
5.404.1 Detailed Description	4189
5.405 opt_random.h File Reference	4189
5.405.1 Detailed Description	4189
5.406 optional File Reference	4189
5.406.1 Detailed Description	4189
5.407 optional File Reference	4189
5.407.1 Detailed Description	4190
5.408 order_statistics_imp.hpp File Reference	4190
5.408.1 Detailed Description	4190
5.409 order_statistics_imp.hpp File Reference	4190
5.409.1 Detailed Description	4190
5.410 os_defines.h File Reference	4190
5.410.1 Detailed Description	4190
5.411 ostream File Reference	4191
5.411.1 Detailed Description	4192
5.412 ostream.tcc File Reference	4192
5.412.1 Detailed Description	4193
5.413 ostream_insert.h File Reference	4193

5.413.1 Detailed Description	4193
5.414 ov_tree_map.hpp File Reference	4193
5.414.1 Detailed Description	4194
5.415 pairing_heap.hpp File Reference	4194
5.415.1 Detailed Description	4194
5.416 par_loop.h File Reference	4194
5.416.1 Detailed Description	4194
5.417 parallel.h File Reference	4195
5.417.1 Detailed Description	4195
5.418 parse_numbers.h File Reference	4195
5.418.1 Detailed Description	4195
5.419 partial_sum.h File Reference	4195
5.419.1 Detailed Description	4196
5.420 partition.h File Reference	4196
5.420.1 Detailed Description	4196
5.420.2 Macro Definition Documentation	4196
5.421 pat_trie.hpp File Reference	4197
5.421.1 Detailed Description	4197
5.422 pat_trie_base.hpp File Reference	4197
5.422.1 Detailed Description	4198
5.423 pod_char_traits.h File Reference	4198
5.423.1 Detailed Description	4198
5.424 point_const_iterator.hpp File Reference	4198
5.424.1 Detailed Description	4199
5.425 point_const_iterator.hpp File Reference	4199
5.425.1 Detailed Description	4199
5.426 point_const_iterator.hpp File Reference	4199
5.426.1 Detailed Description	4199
5.427 point_iterator.hpp File Reference	4199
5.427.1 Detailed Description	4199
5.428 point_iterators.hpp File Reference	4200
5.428.1 Detailed Description	4200
5.429 pointer.h File Reference	4200
5.429.1 Detailed Description	4202
5.430 policy_access_fn_imps.hpp File Reference	4202
5.430.1 Detailed Description	4202
5.431 policy_access_fn_imps.hpp File Reference	4202
5.431.1 Detailed Description	4202
5.432 policy_access_fn_imps.hpp File Reference	4202

5.432.1 Detailed Description	4202
5.433 policy_access_fn_imps.hpp File Reference	4203
5.433.1 Detailed Description	4203
5.434 policy_access_fn_imps.hpp File Reference	4203
5.434.1 Detailed Description	4203
5.435 policy_access_fn_imps.hpp File Reference	4203
5.435.1 Detailed Description	4203
5.436 policy_access_fn_imps.hpp File Reference	4203
5.436.1 Detailed Description	4203
5.437 pool_allocator.h File Reference	4203
5.437.1 Detailed Description	4204
5.438 postypes.h File Reference	4204
5.438.1 Detailed Description	4204
5.439 predefined_ops.h File Reference	4204
5.439.1 Detailed Description	4205
5.440 prefix_search_node_update_imp.hpp File Reference	4205
5.440.1 Detailed Description	4205
5.441 priority_queue.hpp File Reference	4206
5.441.1 Detailed Description	4206
5.442 priority_queue_base_dispatch.hpp File Reference	4206
5.442.1 Detailed Description	4206
5.443 probe_fn_base.hpp File Reference	4206
5.443.1 Detailed Description	4207
5.444 propagate_const File Reference	4207
5.444.1 Detailed Description	4209
5.445 ptr_traits.h File Reference	4209
5.445.1 Detailed Description	4209
5.446 quadratic_probe_fn_imp.hpp File Reference	4209
5.446.1 Detailed Description	4209
5.447 queue File Reference	4210
5.447.1 Detailed Description	4210
5.448 queue.h File Reference	4210
5.448.1 Detailed Description	4210
5.448.2 Macro Definition Documentation	4210
5.449 quicksort.h File Reference	4211
5.449.1 Detailed Description	4211
5.450 quoted_string.h File Reference	4211
5.450.1 Detailed Description	4212
5.451 r_erase_fn_imps.hpp File Reference	4212

5.451.1 Detailed Description	4212
5.452 r_erase_fn_imps.hpp File Reference	4212
5.452.1 Detailed Description	4212
5.453 random File Reference	4212
5.453.1 Detailed Description	4212
5.454 random File Reference	4212
5.454.1 Detailed Description	4213
5.455 random.h File Reference	4213
5.455.1 Detailed Description	4217
5.456 random.tcc File Reference	4217
5.456.1 Detailed Description	4220
5.457 random.tcc File Reference	4221
5.457.1 Detailed Description	4223
5.458 random_number.h File Reference	4223
5.458.1 Detailed Description	4223
5.459 random_shuffle.h File Reference	4223
5.459.1 Detailed Description	4224
5.460 range_access.h File Reference	4224
5.460.1 Detailed Description	4225
5.461 range_cmp.h File Reference	4226
5.461.1 Detailed Description	4226
5.462 ranged_hash_fn.hpp File Reference	4226
5.462.1 Detailed Description	4226
5.463 ranged_probe_fn.hpp File Reference	4226
5.463.1 Detailed Description	4227
5.464 ranges File Reference	4227
5.464.1 Detailed Description	4227
5.465 ranges_algo.h File Reference	4227
5.465.1 Detailed Description	4227
5.466 ranges_algobase.h File Reference	4227
5.466.1 Detailed Description	4227
5.467 ranges_uninitialized.h File Reference	4228
5.467.1 Detailed Description	4228
5.468 ratio File Reference	4228
5.468.1 Detailed Description	4229
5.469 ratio File Reference	4229
5.469.1 Detailed Description	4229
5.470 ratio File Reference	4229
5.470.1 Detailed Description	4229

5.471 rb_tree File Reference	4230
5.471.1 Detailed Description	4230
5.472 rb_tree.hpp File Reference	4230
5.472.1 Detailed Description	4230
5.473 rc.hpp File Reference	4231
5.473.1 Detailed Description	4231
5.474 rc_binomial_heap.hpp File Reference	4231
5.474.1 Detailed Description	4231
5.475 rc_string_base.h File Reference	4231
5.475.1 Detailed Description	4232
5.476 refwrap.h File Reference	4232
5.476.1 Detailed Description	4232
5.477 regex File Reference	4232
5.477.1 Detailed Description	4232
5.478 regex File Reference	4232
5.478.1 Detailed Description	4233
5.479 regex.h File Reference	4233
5.479.1 Detailed Description	4235
5.480 regex.tcc File Reference	4235
5.480.1 Detailed Description	4236
5.481 regex_automaton.h File Reference	4236
5.481.1 Detailed Description	4236
5.482 regex_automaton.tcc File Reference	4237
5.482.1 Detailed Description	4237
5.483 regex_compiler.h File Reference	4237
5.483.1 Detailed Description	4238
5.484 regex_compiler.tcc File Reference	4238
5.484.1 Detailed Description	4238
5.485 regex_constants.h File Reference	4238
5.485.1 Detailed Description	4239
5.486 regex_error.h File Reference	4240
5.486.1 Detailed Description	4240
5.487 regex_executor.h File Reference	4241
5.487.1 Detailed Description	4241
5.488 regex_executor.tcc File Reference	4241
5.488.1 Detailed Description	4241
5.489 regex_scanner.h File Reference	4241
5.489.1 Detailed Description	4241
5.490 regex_scanner.tcc File Reference	4242

5.490.1 Detailed Description	4242
5.491 resize_fn_imps.hpp File Reference	4242
5.491.1 Detailed Description	4242
5.492 resize_fn_imps.hpp File Reference	4242
5.492.1 Detailed Description	4242
5.493 resize_no_store_hash_fn_imps.hpp File Reference	4242
5.493.1 Detailed Description	4242
5.494 resize_no_store_hash_fn_imps.hpp File Reference	4242
5.494.1 Detailed Description	4242
5.495 resize_policy.hpp File Reference	4242
5.495.1 Detailed Description	4243
5.496 resize_store_hash_fn_imps.hpp File Reference	4243
5.496.1 Detailed Description	4243
5.497 resize_store_hash_fn_imps.hpp File Reference	4243
5.497.1 Detailed Description	4243
5.498 rope File Reference	4243
5.498.1 Detailed Description	4246
5.499 ropeimpl.h File Reference	4246
5.499.1 Detailed Description	4247
5.500 rotate_fn_imps.hpp File Reference	4247
5.500.1 Detailed Description	4247
5.501 rotate_fn_imps.hpp File Reference	4247
5.501.1 Detailed Description	4247
5.502 safe_base.h File Reference	4247
5.502.1 Detailed Description	4248
5.503 safe_container.h File Reference	4248
5.503.1 Detailed Description	4248
5.504 safe_iterator.h File Reference	4248
5.504.1 Detailed Description	4249
5.505 safe_iterator.tcc File Reference	4249
5.505.1 Detailed Description	4250
5.506 safe_local_iterator.h File Reference	4250
5.506.1 Detailed Description	4251
5.507 safe_local_iterator.tcc File Reference	4251
5.507.1 Detailed Description	4251
5.508 safe_sequence.h File Reference	4251
5.508.1 Detailed Description	4251
5.509 safe_sequence.tcc File Reference	4251
5.509.1 Detailed Description	4252

5.510 safe_unordered_base.h File Reference	4252
5.510.1 Detailed Description	4252
5.511 safe_unordered_container.h File Reference	4252
5.511.1 Detailed Description	4252
5.512 safe_unordered_container.tcc File Reference	4252
5.512.1 Detailed Description	4253
5.513 sample_probe_fn.hpp File Reference	4253
5.513.1 Detailed Description	4253
5.514 sample_range_hashing.hpp File Reference	4253
5.514.1 Detailed Description	4253
5.515 sample_ranged_hash_fn.hpp File Reference	4253
5.515.1 Detailed Description	4254
5.516 sample_ranged_probe_fn.hpp File Reference	4254
5.516.1 Detailed Description	4254
5.517 sample_resize_policy.hpp File Reference	4254
5.517.1 Detailed Description	4254
5.518 sample_resize_trigger.hpp File Reference	4254
5.518.1 Detailed Description	4255
5.519 sample_size_policy.hpp File Reference	4255
5.519.1 Detailed Description	4255
5.520 sample_tree_node_update.hpp File Reference	4255
5.520.1 Detailed Description	4255
5.521 sample_trie_access_traits.hpp File Reference	4255
5.521.1 Detailed Description	4256
5.522 sample_trie_node_update.hpp File Reference	4256
5.522.1 Detailed Description	4256
5.523 sample_update_policy.hpp File Reference	4256
5.523.1 Detailed Description	4256
5.524 scoped_allocator File Reference	4256
5.524.1 Detailed Description	4257
5.525 search.h File Reference	4257
5.525.1 Detailed Description	4257
5.526 set File Reference	4257
5.526.1 Detailed Description	4257
5.527 set File Reference	4258
5.527.1 Detailed Description	4258
5.528 set File Reference	4258
5.528.1 Detailed Description	4259
5.529 set.h File Reference	4259

5.529.1 Detailed Description	4260
5.530 set_operations.h File Reference	4260
5.530.1 Detailed Description	4260
5.531 settings.h File Reference	4260
5.531.1 Detailed Description	4261
5.531.2 Deciding whether to run an algorithm in parallel.	4261
5.531.3 Macro Definition Documentation	4261
5.532 shared_mutex File Reference	4262
5.532.1 Detailed Description	4262
5.533 shared_ptr.h File Reference	4262
5.533.1 Detailed Description	4263
5.534 shared_ptr.h File Reference	4263
5.534.1 Detailed Description	4265
5.534.2 Function Documentation	4265
5.535 shared_ptr_atomic.h File Reference	4266
5.535.1 Detailed Description	4266
5.536 shared_ptr_base.h File Reference	4266
5.536.1 Detailed Description	4268
5.537 size_fn_imps.hpp File Reference	4268
5.537.1 Detailed Description	4268
5.538 slice_array.h File Reference	4268
5.538.1 Detailed Description	4268
5.539 slist File Reference	4268
5.539.1 Detailed Description	4269
5.540 sort.h File Reference	4269
5.540.1 Detailed Description	4270
5.541 span File Reference	4270
5.541.1 Detailed Description	4270
5.542 specfun.h File Reference	4270
5.542.1 Detailed Description	4273
5.543 splay_fn_imps.hpp File Reference	4273
5.543.1 Detailed Description	4273
5.544 splay_tree_.hpp File Reference	4273
5.544.1 Detailed Description	4273
5.545 split_fn_imps.hpp File Reference	4274
5.545.1 Detailed Description	4274
5.546 split_join_fn_imps.hpp File Reference	4274
5.546.1 Detailed Description	4274
5.547 split_join_fn_imps.hpp File Reference	4274

5.547.1 Detailed Description	4274
5.548 split_join_fn_imps.hpp File Reference	4274
5.548.1 Detailed Description	4274
5.549 split_join_fn_imps.hpp File Reference	4274
5.549.1 Detailed Description	4274
5.550 split_join_fn_imps.hpp File Reference	4274
5.550.1 Detailed Description	4274
5.551 split_join_fn_imps.hpp File Reference	4275
5.551.1 Detailed Description	4275
5.552 split_join_fn_imps.hpp File Reference	4275
5.552.1 Detailed Description	4275
5.553 split_join_fn_imps.hpp File Reference	4275
5.553.1 Detailed Description	4275
5.554 split_join_fn_imps.hpp File Reference	4275
5.554.1 Detailed Description	4275
5.555 sso_string_base.h File Reference	4275
5.555.1 Detailed Description	4275
5.556 sstream File Reference	4275
5.556.1 Detailed Description	4276
5.557 sstream.tcc File Reference	4276
5.557.1 Detailed Description	4276
5.558 stack File Reference	4277
5.558.1 Detailed Description	4277
5.559 standard_policies.hpp File Reference	4277
5.559.1 Detailed Description	4277
5.559.2 Enumeration Type Documentation	4277
5.560 std_abs.h File Reference	4278
5.560.1 Detailed Description	4278
5.561 std_function.h File Reference	4278
5.561.1 Detailed Description	4279
5.562 std_mutex.h File Reference	4279
5.562.1 Detailed Description	4279
5.563 stdc++.h File Reference	4280
5.563.1 Detailed Description	4280
5.564 stdexcept File Reference	4280
5.564.1 Detailed Description	4280
5.565 stdio_filebuf.h File Reference	4280
5.565.1 Detailed Description	4281
5.566 stdio_sync_filebuf.h File Reference	4281

5.566.1 Detailed Description	4281
5.567 stdlib.h File Reference	4281
5.567.1 Detailed Description	4281
5.568 stdtr1c++.h File Reference	4281
5.568.1 Detailed Description	4281
5.569 stl_algo.h File Reference	4281
5.569.1 Detailed Description	4291
5.569.2 Function Documentation	4291
5.570 stl_algobase.h File Reference	4292
5.570.1 Detailed Description	4297
5.571 stl_bvector.h File Reference	4297
5.571.1 Detailed Description	4298
5.572 stl_construct.h File Reference	4298
5.572.1 Detailed Description	4298
5.573 stl_deque.h File Reference	4299
5.573.1 Detailed Description	4299
5.573.2 Macro Definition Documentation	4299
5.574 stl_function.h File Reference	4300
5.574.1 Detailed Description	4302
5.575 stl_heap.h File Reference	4302
5.575.1 Detailed Description	4303
5.576 stl_iterator.h File Reference	4303
5.576.1 Detailed Description	4307
5.577 stl_iterator.h File Reference	4307
5.577.1 Detailed Description	4308
5.578 stl_iterator_base_funcs.h File Reference	4308
5.578.1 Detailed Description	4309
5.579 stl_iterator_base_types.h File Reference	4309
5.579.1 Detailed Description	4309
5.580 stl_list.h File Reference	4310
5.580.1 Detailed Description	4310
5.581 stl_map.h File Reference	4310
5.581.1 Detailed Description	4311
5.582 stl_multimap.h File Reference	4311
5.582.1 Detailed Description	4312
5.583 stl_multiset.h File Reference	4312
5.583.1 Detailed Description	4313
5.584 stl_numeric.h File Reference	4313
5.584.1 Detailed Description	4314

5.585 stl_pair.h File Reference	4314
5.585.1 Detailed Description	4315
5.586 stl_queue.h File Reference	4315
5.586.1 Detailed Description	4315
5.587 stl_raw_storage_iter.h File Reference	4315
5.587.1 Detailed Description	4316
5.588 stl_relops.h File Reference	4316
5.588.1 Detailed Description	4316
5.589 stl_set.h File Reference	4316
5.589.1 Detailed Description	4317
5.590 stl_stack.h File Reference	4317
5.590.1 Detailed Description	4318
5.591 stl_tempbuf.h File Reference	4318
5.591.1 Detailed Description	4318
5.592 stl_tree.h File Reference	4318
5.592.1 Detailed Description	4319
5.593 stl_uninitialized.h File Reference	4319
5.593.1 Detailed Description	4320
5.594 stl_vector.h File Reference	4320
5.594.1 Detailed Description	4320
5.595 stop_token File Reference	4321
5.595.1 Detailed Description	4321
5.596 stream_iterator.h File Reference	4321
5.596.1 Detailed Description	4321
5.597 streambuf File Reference	4321
5.597.1 Detailed Description	4322
5.598 streambuf.tcc File Reference	4322
5.598.1 Detailed Description	4322
5.599 streambuf_iterator.h File Reference	4322
5.599.1 Detailed Description	4323
5.600 string File Reference	4323
5.600.1 Detailed Description	4323
5.601 string File Reference	4324
5.601.1 Detailed Description	4326
5.602 string File Reference	4326
5.602.1 Detailed Description	4326
5.603 string_conversions.h File Reference	4326
5.603.1 Detailed Description	4327
5.604 string_view File Reference	4327

5.604.1 Detailed Description	4329
5.605 string_view.tcc File Reference	4329
5.605.1 Detailed Description	4329
5.606 string_view.tcc File Reference	4329
5.606.1 Detailed Description	4330
5.607 stringfwd.h File Reference	4330
5.607.1 Detailed Description	4330
5.608 stringstream File Reference	4330
5.608.1 Detailed Description	4330
5.609 synth_access_traits.hpp File Reference	4330
5.609.1 Detailed Description	4331
5.610 system_error File Reference	4331
5.610.1 Detailed Description	4332
5.611 system_error File Reference	4332
5.611.1 Detailed Description	4332
5.612 tag_and_trait.hpp File Reference	4332
5.612.1 Detailed Description	4333
5.613 tags.h File Reference	4334
5.613.1 Detailed Description	4334
5.614 tgmth.h File Reference	4334
5.614.1 Detailed Description	4334
5.615 thin_heap.hpp File Reference	4334
5.615.1 Detailed Description	4335
5.616 thread File Reference	4335
5.616.1 Detailed Description	4336
5.617 throw_allocator.h File Reference	4336
5.617.1 Detailed Description	4337
5.618 time_members.h File Reference	4337
5.618.1 Detailed Description	4338
5.619 trace_fn_imps.hpp File Reference	4338
5.619.1 Detailed Description	4338
5.620 trace_fn_imps.hpp File Reference	4338
5.620.1 Detailed Description	4338
5.621 trace_fn_imps.hpp File Reference	4338
5.621.1 Detailed Description	4338
5.622 trace_fn_imps.hpp File Reference	4338
5.622.1 Detailed Description	4338
5.623 trace_fn_imps.hpp File Reference	4338
5.623.1 Detailed Description	4338

5.624	trace_fn_imps.hpp File Reference	4338
5.624.1	Detailed Description	4338
5.625	trace_fn_imps.hpp File Reference	4339
5.625.1	Detailed Description	4339
5.626	trace_fn_imps.hpp File Reference	4339
5.626.1	Detailed Description	4339
5.627	traits.hpp File Reference	4339
5.627.1	Detailed Description	4339
5.628	traits.hpp File Reference	4339
5.628.1	Detailed Description	4340
5.629	traits.hpp File Reference	4340
5.629.1	Detailed Description	4340
5.630	traits.hpp File Reference	4340
5.630.1	Detailed Description	4340
5.631	traits.hpp File Reference	4340
5.631.1	Detailed Description	4341
5.632	traits.hpp File Reference	4341
5.632.1	Detailed Description	4341
5.633	tree_policy.hpp File Reference	4341
5.633.1	Detailed Description	4341
5.634	tree_trace_base.hpp File Reference	4342
5.634.1	Detailed Description	4342
5.635	trie_policy.hpp File Reference	4342
5.635.1	Detailed Description	4342
5.636	trie_policy_base.hpp File Reference	4342
5.636.1	Detailed Description	4343
5.637	trie_string_access_traits_imp.hpp File Reference	4343
5.637.1	Detailed Description	4343
5.638	tuple File Reference	4343
5.638.1	Detailed Description	4345
5.639	tuple File Reference	4345
5.639.1	Detailed Description	4345
5.640	type_traits File Reference	4346
5.640.1	Detailed Description	4351
5.640.2	Macro Definition Documentation	4351
5.641	type_traits File Reference	4351
5.641.1	Detailed Description	4352
5.642	type_traits File Reference	4352
5.642.1	Detailed Description	4355

5.643	type_traits.h File Reference	4355
5.643.1	Detailed Description	4356
5.644	type_utils.hpp File Reference	4356
5.644.1	Detailed Description	4356
5.645	typeid File Reference	4356
5.645.1	Detailed Description	4357
5.646	typeid File Reference	4357
5.646.1	Detailed Description	4357
5.647	typelist.h File Reference	4357
5.647.1	Detailed Description	4358
5.648	types.h File Reference	4359
5.648.1	Detailed Description	4359
5.649	types_traits.hpp File Reference	4359
5.649.1	Detailed Description	4360
5.650	uniform_int_dist.h File Reference	4360
5.650.1	Detailed Description	4360
5.651	unique_copy.h File Reference	4360
5.651.1	Detailed Description	4361
5.652	unique_lock.h File Reference	4361
5.652.1	Detailed Description	4361
5.653	unique_ptr.h File Reference	4361
5.653.1	Detailed Description	4362
5.654	unordered_map File Reference	4362
5.654.1	Detailed Description	4362
5.655	unordered_map File Reference	4362
5.655.1	Detailed Description	4363
5.656	unordered_map File Reference	4363
5.656.1	Detailed Description	4363
5.657	unordered_map.h File Reference	4363
5.657.1	Detailed Description	4364
5.658	unordered_set File Reference	4365
5.658.1	Detailed Description	4365
5.659	unordered_set File Reference	4365
5.659.1	Detailed Description	4366
5.660	unordered_set File Reference	4366
5.660.1	Detailed Description	4366
5.661	unordered_set.h File Reference	4366
5.661.1	Detailed Description	4367
5.662	update_fn_imps.hpp File Reference	4367

5.662.1 Detailed Description	4367
5.663 utility File Reference	4368
5.663.1 Detailed Description	4369
5.664 utility File Reference	4369
5.664.1 Detailed Description	4370
5.665 valarray File Reference	4370
5.665.1 Detailed Description	4374
5.666 valarray_after.h File Reference	4374
5.666.1 Detailed Description	4384
5.667 valarray_array.h File Reference	4384
5.667.1 Detailed Description	4392
5.668 valarray_array.tcc File Reference	4392
5.668.1 Detailed Description	4393
5.669 valarray_before.h File Reference	4393
5.669.1 Detailed Description	4393
5.670 variant File Reference	4393
5.670.1 Detailed Description	4393
5.671 vector File Reference	4393
5.671.1 Detailed Description	4394
5.672 vector File Reference	4394
5.672.1 Detailed Description	4394
5.673 vector File Reference	4395
5.673.1 Detailed Description	4395
5.674 vector.tcc File Reference	4395
5.674.1 Detailed Description	4395
5.675 vstring.h File Reference	4396
5.675.1 Detailed Description	4398
5.676 vstring.tcc File Reference	4398
5.676.1 Detailed Description	4399
5.677 vstring_fwd.h File Reference	4399
5.677.1 Detailed Description	4400
5.678 vstring_util.h File Reference	4400
5.678.1 Detailed Description	4400
5.679 workstealing.h File Reference	4400
5.679.1 Detailed Description	4401

1 Todo List

Member [__gnu_cxx::distance](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [_Distance](#) &__n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::hash_map](#)<[_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::hash_multimap](#)<[_Key](#), [_Tp](#), [_HashFn](#), [_EqualKey](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::hash_multiset](#)<[_Value](#), [_HashFcn](#), [_EqualKey](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::hash_set](#)<[_Value](#), [_HashFcn](#), [_EqualKey](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::power](#) ([_Tp](#) __x, [_Integer](#) __n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::power](#) ([_Tp](#) __x, [_Integer](#) __n, [_MonoidOperation](#) __monoid_op)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::random_sample](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [_RandomAccessIterator](#) __↵
out_first, [_RandomAccessIterator](#) __out_last, [_RandomNumberGenerator](#) &__rand)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::random_sample](#) ([_InputIterator](#) __first, [_InputIterator](#) __last, [_RandomAccessIterator](#) __↵
out_first, [_RandomAccessIterator](#) __out_last)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::random_sample_n](#) ([_ForwardIterator](#) __first, [_ForwardIterator](#) __last, [_OutputIterator](#) __↵
out, const [_Distance](#) __n, [_RandomNumberGenerator](#) &__rand)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Member [__gnu_cxx::random_sample_n](#) ([_ForwardIterator](#) __first, [_ForwardIterator](#) __last, [_OutputIterator](#) __↵
out, const [_Distance](#) __n)

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::rb_tree](#)<[_Key](#), [_Value](#), [_KeyOfValue](#), [_Compare](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class [__gnu_cxx::rope](#)<[_CharT](#), [_Alloc](#) >

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Class `__gnu_cxx::slist<_Tp, _Alloc>`

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Module `mathsf`

Provide accuracy comparisons on a per-function basis for a small number of targets.

Class `std::basic_string<_CharT, _Traits, _Alloc>`

Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

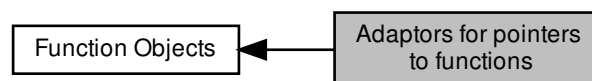
Member `std::regex_traits<_Ch_type>::transform_primary(_Fwd_iter __first, _Fwd_iter __last) const`

Implement this function correctly.

2 Module Documentation

2.1 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class `std::pointer_to_binary_function<_Arg1, _Arg2, _Result>`
- class `std::pointer_to_unary_function<_Arg, _Result>`

Functions

- `template<typename _Arg, typename _Result>`
`pointer_to_unary_function<_Arg, _Result> std::ptr_fun(_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result>`
`pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun(_Result(*__x)(_Arg1, _Arg2))`

2.1.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.

Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.

The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

2.1.2 Function Documentation

2.1.2.1 `ptr_fun()` [1/2]

```
template<typename _Arg , typename _Result >
pointer_to_unary_function<_Arg, _Result> std::ptr_fun (
    _Result(*) (_Arg) __x ) [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 1100 of file `stl_function.h`.

2.1.2.2 `ptr_fun()` [2/2]

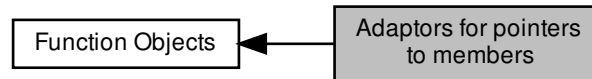
```
template<typename _Arg1 , typename _Arg2 , typename _Result >
pointer_to_binary_function<_Arg1, _Arg2, _Result> std::ptr_fun (
    _Result(*) (_Arg1, _Arg2) __x ) [inline]
```

One of the [adaptors for function pointers](#).

Definition at line 1126 of file `stl_function.h`.

2.2 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::const_mem_fun_ref_t<_Ret, _Tp>`
- class `std::const_mem_fun_t<_Ret, _Tp>`
- class `std::mem_fun1_ref_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun1_t<_Ret, _Tp, _Arg>`
- class `std::mem_fun_ref_t<_Ret, _Tp>`
- class `std::mem_fun_t<_Ret, _Tp>`

Functions

- `template<typename _Ret, typename _Tp>`
`mem_fun_t<_Ret, _Tp> std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_t<_Ret, _Tp, _Arg> std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp>`
`mem_fun_ref_t<_Ret, _Tp> std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp, typename _Arg>`
`mem_fun1_ref_t<_Ret, _Tp, _Arg> std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`

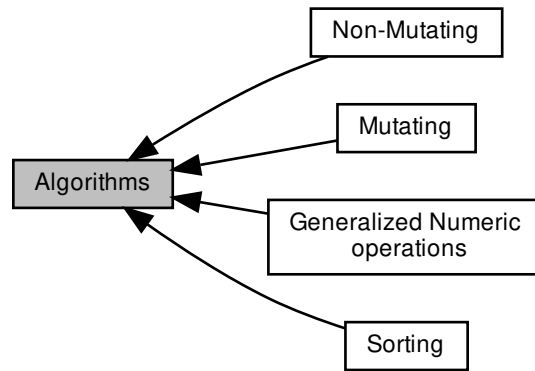
2.2.1 Detailed Description

There are a total of $8 = 2^3$ function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.

All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

2.3 Algorithms

Collaboration diagram for Algorithms:



Modules

- [Generalized Numeric operations](#)
- [Mutating](#)
- [Non-Mutating](#)
- [Sorting](#)

2.3.1 Detailed Description

Components for performing algorithmic operations. Includes non-modifying sequence, modifying (mutating) sequence, sorting, searching, merge, partition, heap, set, minima, maxima, and permutation operations.

2.4 Allocators

Collaboration diagram for Allocators:



Files

- file [scoped_allocator](#)

Classes

- struct [__gnu_cxx::__alloc_traits](#)< [_Alloc](#), [typename](#) >
- class [__gnu_cxx::__mt_alloc](#)< [_Tp](#), [_Poolp](#) >
- class [__gnu_cxx::__pool_alloc](#)< [_Tp](#) >
- class [__gnu_cxx::__ExtPtr_allocator](#)< [_Tp](#) >
- class [std::allocator](#)< [_Tp](#) >
- class [std::allocator](#)< [void](#) >
- struct [std::allocator_traits](#)< [_Alloc](#) >
- class [__gnu_cxx::bitmap_allocator](#)< [_Tp](#) >
- class [__gnu_cxx::debug_allocator](#)< [_Alloc](#) >
- class [__gnu_cxx::malloc_allocator](#)< [_Tp](#) >
- class [__gnu_cxx::new_allocator](#)< [_Tp](#) >
- class [std::scoped_allocator_adaptor](#)< [_OuterAlloc](#), [_InnerAllocs](#) >
- class [__gnu_cxx::throw_allocator_base](#)< [_Tp](#), [_Cond](#) >
- struct [std::uses_allocator](#)< [_Tp](#), [_Alloc](#) >

Typedefs

- [template](#)<[typename](#) [_Tp](#) >
using [std::__allocator_base](#) = [__gnu_cxx::new_allocator](#)< [_Tp](#) >

Functions

- [template](#)<[typename](#) [_T1](#), [typename](#) [_T2](#) >
constexpr bool **std::operator!=** (const [allocator](#)< [_T1](#) > &, const [allocator](#)< [_T2](#) > &) noexcept
- [template](#)<[typename](#) [_OutA1](#), [typename](#) [_OutA2](#), [typename](#)... [_InA](#)>
bool **operator!=** (const [scoped_allocator_adaptor](#)< [_OutA1](#), [_InA](#)... > &__a, const [scoped_allocator_adaptor](#)< [_OutA2](#), [_InA](#)... > &__b) noexcept
- [template](#)<[typename](#) [_T1](#), [typename](#) [_T2](#) >
constexpr bool **std::operator==** (const [allocator](#)< [_T1](#) > &, const [allocator](#)< [_T2](#) > &) noexcept
- [template](#)<[typename](#) [_OutA1](#), [typename](#) [_OutA2](#), [typename](#)... [_InA](#)>
bool **operator==** (const [scoped_allocator_adaptor](#)< [_OutA1](#), [_InA](#)... > &__a, const [scoped_allocator_adaptor](#)< [_OutA2](#), [_InA](#)... > &__b) noexcept

2.4.1 Detailed Description

Classes encapsulating memory operations.

2.4.2 Typedef Documentation

2.4.2.1 `__allocator_base`

```
template<typename _Tp >
using std::__allocator_base = typedef __gnu_cxx::new_allocator<_Tp>
```

An alias to the base class for `std::allocator`.

Used to set the `std::allocator` base class to `__gnu_cxx::new_allocator`.

Template Parameters

<code>_Tp</code>	Type of allocated object.
------------------	---------------------------

Definition at line 48 of file `c++allocator.h`.

2.4.3 Function Documentation

2.4.3.1 `operator!=()`

```
template<typename _OutA1 , typename _OutA2 , typename... _InA>
bool operator!=(
    const scoped_allocator_adaptor< _OutA1, _InA... > & __a,
    const scoped_allocator_adaptor< _OutA2, _InA... > & __b ) [related]
```

Definition at line 507 of file `scoped_allocator`.

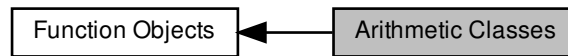
2.4.3.2 `operator==()`

```
template<typename _OutA1 , typename _OutA2 , typename... _InA>
bool operator==(
    const scoped_allocator_adaptor< _OutA1, _InA... > & __a,
    const scoped_allocator_adaptor< _OutA2, _InA... > & __b ) [related]
```

Definition at line 496 of file `scoped_allocator`.

2.5 Arithmetic Classes

Collaboration diagram for Arithmetic Classes:



Classes

- struct `std::divides< _Tp >`
- struct `std::divides< void >`
- struct `std::minus< _Tp >`
- struct `std::minus< void >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< void >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< void >`
- struct `std::negate< _Tp >`
- struct `std::negate< void >`
- struct `std::plus< _Tp >`

Macros

- `#define __cpp_lib_transparent_operators`

2.5.1 Detailed Description

Because basic math often needs to be done during an algorithm, the library provides functors for those operations. See the documentation for [the base classes](#) for examples of their use.

2.6 Array creation functions

Collaboration diagram for Array creation functions:



Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>`
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::__to_array (_Tp(&__a)[_Nm], index_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>`
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::fundamentals_v2::make_array (_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>`
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (_Tp(&__a)[_Nm])`
`noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)`

2.6.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

2.6.2 Function Documentation

2.6.2.1 `make_array()`

```

template<typename _Dest = void, typename... _Types>
constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...( _Types)> std::experimental::fundamentals\_v2::make\_array (
    _Types &&... __t )
  
```

Create a `std::array` from a variable-length list of arguments.

Definition at line 86 of file `experimental/array`.

2.6.2.2 `to_array()`

```

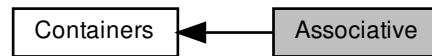
template<typename _Tp, size_t _Nm>
constexpr array< remove\_cv\_t< _Tp >, _Nm > std::experimental::fundamentals\_v2::to\_array (
    _Tp(& __a[_Nm] ) ) [noexcept]
  
```

Create a `std::array` from an array.

Definition at line 101 of file `experimental/array`.

2.7 Associative

Collaboration diagram for Associative:



Classes

- class `std::map<_Key, _Tp, _Compare, _Alloc >`
- class `std::multimap<_Key, _Tp, _Compare, _Alloc >`
- class `std::multiset<_Key, _Compare, _Alloc >`
- class `std::set<_Key, _Compare, _Alloc >`

2.7.1 Detailed Description

Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in [tables](#).

2.8 Atomics

Classes

- struct `std::__atomic_base<_ITp>`
- struct `std::__atomic_base<_PTp*>`
- struct `std::__atomic_flag_base`
- struct `std::atomic<_Tp>`
- struct `std::atomic<_Tp*>`
- struct `std::atomic<bool>`
- struct `std::atomic<char>`
- struct `std::atomic<char16_t>`
- struct `std::atomic<char32_t>`
- struct `std::atomic<int>`
- struct `std::atomic<long>`
- struct `std::atomic<long long>`
- struct `std::atomic<short>`
- struct `std::atomic<signed char>`
- struct `std::atomic<unsigned char>`
- struct `std::atomic<unsigned int>`
- struct `std::atomic<unsigned long>`
- struct `std::atomic<unsigned long long>`
- struct `std::atomic<unsigned short>`
- struct `std::atomic<wchar_t>`
- struct `std::atomic_flag`

Macros

- `#define _GLIBCXX20_INIT(l)`
- `#define _GLIBCXX20_INIT(l)`
- `#define ATOMIC_BOOL_LOCK_FREE`
- `#define ATOMIC_CHAR16_T_LOCK_FREE`
- `#define ATOMIC_CHAR32_T_LOCK_FREE`
- `#define ATOMIC_CHAR_LOCK_FREE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_INT_LOCK_FREE`
- `#define ATOMIC_LLONG_LOCK_FREE`
- `#define ATOMIC_LONG_LOCK_FREE`
- `#define ATOMIC_POINTER_LOCK_FREE`
- `#define ATOMIC_SHORT_LOCK_FREE`
- `#define ATOMIC_VAR_INIT(_Vl)`
- `#define ATOMIC_WCHAR_T_LOCK_FREE`

Typedefs

- `template<typename _Tp >`
`using std::__atomic_diff_t = typename atomic< _Tp >::difference_type`
- `typedef unsigned char std::__atomic_flag_data_type`
- `template<typename _Tp >`
`using std::__atomic_val_t = typename atomic< _Tp >::value_type`
- `typedef atomic< bool > std::atomic_bool`
- `typedef atomic< char > std::atomic_char`
- `typedef atomic< char16_t > std::atomic_char16_t`
- `typedef atomic< char32_t > std::atomic_char32_t`
- `typedef atomic< int > std::atomic_int`
- `typedef atomic< int16_t > std::atomic_int16_t`
- `typedef atomic< int32_t > std::atomic_int32_t`
- `typedef atomic< int64_t > std::atomic_int64_t`
- `typedef atomic< int8_t > std::atomic_int8_t`
- `typedef atomic< int_fast16_t > std::atomic_int_fast16_t`
- `typedef atomic< int_fast32_t > std::atomic_int_fast32_t`
- `typedef atomic< int_fast64_t > std::atomic_int_fast64_t`
- `typedef atomic< int_fast8_t > std::atomic_int_fast8_t`
- `typedef atomic< int_least16_t > std::atomic_int_least16_t`
- `typedef atomic< int_least32_t > std::atomic_int_least32_t`
- `typedef atomic< int_least64_t > std::atomic_int_least64_t`
- `typedef atomic< int_least8_t > std::atomic_int_least8_t`
- `typedef atomic< intmax_t > std::atomic_intmax_t`
- `typedef atomic< intptr_t > std::atomic_intptr_t`
- `typedef atomic< long long > std::atomic_llong`
- `typedef atomic< long > std::atomic_long`
- `typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t`
- `typedef atomic< signed char > std::atomic_schar`
- `typedef atomic< short > std::atomic_short`
- `typedef atomic< size_t > std::atomic_size_t`
- `typedef atomic< unsigned char > std::atomic_uchar`
- `typedef atomic< unsigned int > std::atomic_uint`
- `typedef atomic< uint16_t > std::atomic_uint16_t`
- `typedef atomic< uint32_t > std::atomic_uint32_t`
- `typedef atomic< uint64_t > std::atomic_uint64_t`
- `typedef atomic< uint8_t > std::atomic_uint8_t`
- `typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t`
- `typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t`
- `typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t`
- `typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t`
- `typedef atomic< uint_least16_t > std::atomic_uint_least16_t`
- `typedef atomic< uint_least32_t > std::atomic_uint_least32_t`
- `typedef atomic< uint_least64_t > std::atomic_uint_least64_t`
- `typedef atomic< uint_least8_t > std::atomic_uint_least8_t`
- `typedef atomic< uintmax_t > std::atomic_uintmax_t`
- `typedef atomic< uintptr_t > std::atomic_uintptr_t`
- `typedef atomic< unsigned long long > std::atomic_ullong`
- `typedef atomic< unsigned long > std::atomic_ulong`
- `typedef atomic< unsigned short > std::atomic_ushort`
- `typedef atomic< wchar_t > std::atomic_wchar_t`
- `typedef enum std::memory_order std::memory_order`

Enumerations

- enum `__memory_order_modifier` { `__memory_order_mask`, `__memory_order_modifier_mask`, `__memory_order_hle_acquire`, `__memory_order_hle_release` }
- enum `std::memory_order` { `memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`, `memory_order_release`, `memory_order_acq_rel`, `memory_order_seq_cst` }

Functions

- constexpr `memory_order` `std::__cmpexch_failure_order` (`memory_order` __m) noexcept
- constexpr `memory_order` `std::__cmpexch_failure_order2` (`memory_order` __m) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong_explicit` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, `memory_order` __m1, `memory_order` __m2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_strong_explicit` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, `memory_order` __m1, `memory_order` __m2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak_explicit` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, `memory_order` __m1, `memory_order` __m2) noexcept
- template<typename _ITp >
bool `std::atomic_compare_exchange_weak_explicit` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, `memory_order` __m1, `memory_order` __m2) noexcept
- template<typename _ITp >
_ITp `std::atomic_exchange` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp >
_ITp `std::atomic_exchange` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp >
_ITp `std::atomic_exchange_explicit` (`atomic`< _ITp > *__a, __atomic_val_t< _ITp > __i, `memory_order` __m) noexcept
- template<typename _ITp >
_ITp `std::atomic_exchange_explicit` (volatile `atomic`< _ITp > *__a, __atomic_val_t< _ITp > __i, `memory_order` __m) noexcept
- template<typename _ITp >
_ITp `std::atomic_fetch_add` (`atomic`< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp >
_ITp `std::atomic_fetch_add` (volatile `atomic`< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept

- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m)`
`noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_add_explicit (volatile atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub (volatile atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_sub_explicit (volatile atomic<_ITp> *__a, __atomic_diff_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (__atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`

- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`
`void std::atomic_init (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_init (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const atomic<_ITp> *__a) noexcept`
- `template<typename _ITp >`
`bool std::atomic_is_lock_free (const volatile atomic<_ITp> *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load (const atomic<_ITp> *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load (const volatile atomic<_ITp> *__a) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const atomic<_ITp> *__a, memory_order __m) noexcept`
- `template<typename _ITp >`
`_ITp std::atomic_load_explicit (const volatile atomic<_ITp> *__a, memory_order __m) noexcept`
- `void std::atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp >`
`void std::atomic_store (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_store (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i) noexcept`
- `template<typename _ITp >`
`void std::atomic_store_explicit (atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `template<typename _ITp >`
`void std::atomic_store_explicit (volatile atomic<_ITp> *__a, __atomic_val_t<_ITp> __i, memory_order __m) noexcept`
- `void std::atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Tp >`
`_Tp std::kill_dependency (_Tp __y) noexcept`
- `constexpr memory_order std::operator& (memory_order __m, __memory_order_modifier __mod)`
- `constexpr memory_order std::operator| (memory_order __m, __memory_order_modifier __mod)`

2.8.1 Detailed Description

Components for performing atomic operations.

2.8.2 Macro Definition Documentation

2.8.2.1 ATOMIC_BOOL_LOCK_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

Definition at line 49 of file `atomic_lockfree_defines.h`.

2.8.3 Typedef Documentation

2.8.3.1 atomic_bool

```
typedef atomic<bool> std::atomic_bool
```

atomic_bool

Definition at line 991 of file atomic.

2.8.3.2 atomic_char

```
typedef atomic<char> std::atomic_char
```

atomic_char

Definition at line 994 of file atomic.

2.8.3.3 atomic_char16_t

```
typedef atomic<char16_t> std::atomic_char16_t
```

atomic_char16_t

Definition at line 1035 of file atomic.

2.8.3.4 atomic_char32_t

```
typedef atomic<char32_t> std::atomic_char32_t
```

atomic_char32_t

Definition at line 1038 of file atomic.

2.8.3.5 atomic_int

```
typedef atomic<int> std::atomic_int
```

atomic_int

Definition at line 1009 of file atomic.

2.8.3.6 atomic_int16_t

```
typedef atomic<int16_t> std::atomic_int16_t
```

atomic_int16_t

Definition at line 1051 of file atomic.

2.8.3.7 atomic_int32_t

```
typedef atomic<int32_t> std::atomic_int32_t
```

atomic_int32_t

Definition at line 1057 of file atomic.

2.8.3.8 atomic_int64_t

```
typedef atomic<int64_t> std::atomic_int64_t
```

atomic_int64_t

Definition at line 1063 of file atomic.

2.8.3.9 atomic_int8_t

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic_int8_t

Definition at line 1045 of file atomic.

2.8.3.10 atomic_int_fast16_t

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic_int_fast16_t

Definition at line 1101 of file atomic.

2.8.3.11 `atomic_int_fast32_t`

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

`atomic_int_fast32_t`

Definition at line 1107 of file `atomic`.

2.8.3.12 `atomic_int_fast64_t`

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

`atomic_int_fast64_t`

Definition at line 1113 of file `atomic`.

2.8.3.13 `atomic_int_fast8_t`

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

`atomic_int_fast8_t`

Definition at line 1095 of file `atomic`.

2.8.3.14 `atomic_int_least16_t`

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

`atomic_int_least16_t`

Definition at line 1076 of file `atomic`.

2.8.3.15 `atomic_int_least32_t`

```
typedef atomic<int_least32_t> std::atomic_int_least32_t
```

`atomic_int_least32_t`

Definition at line 1082 of file `atomic`.

2.8.3.16 `atomic_int_least64_t`

```
typedef atomic<int_least64_t> std::atomic_int_least64_t
```

`atomic_int_least64_t`

Definition at line 1088 of file `atomic`.

2.8.3.17 `atomic_int_least8_t`

```
typedef atomic<int_least8_t> std::atomic_int_least8_t
```

`atomic_int_least8_t`

Definition at line 1070 of file `atomic`.

2.8.3.18 `atomic_intmax_t`

```
typedef atomic<intmax_t> std::atomic_intmax_t
```

`atomic_intmax_t`

Definition at line 1134 of file `atomic`.

2.8.3.19 `atomic_intptr_t`

```
typedef atomic<intptr_t> std::atomic_intptr_t
```

`atomic_intptr_t`

Definition at line 1121 of file `atomic`.

2.8.3.20 `atomic_llong`

```
typedef atomic<long long> std::atomic_llong
```

`atomic_llong`

Definition at line 1021 of file `atomic`.

2.8.3.21 atomic_long

```
typedef atomic<long> std::atomic_long
```

atomic_long

Definition at line 1015 of file atomic.

2.8.3.22 atomic_ptrdiff_t

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic_ptrdiff_t

Definition at line 1130 of file atomic.

2.8.3.23 atomic_schar

```
typedef atomic<signed char> std::atomic_schar
```

atomic_schar

Definition at line 997 of file atomic.

2.8.3.24 atomic_short

```
typedef atomic<short> std::atomic_short
```

atomic_short

Definition at line 1003 of file atomic.

2.8.3.25 atomic_size_t

```
typedef atomic<size_t> std::atomic_size_t
```

atomic_size_t

Definition at line 1127 of file atomic.

2.8.3.26 `atomic_uchar`

```
typedef atomic<unsigned char> std::atomic_uchar
```

`atomic_uchar`

Definition at line 1000 of file `atomic`.

2.8.3.27 `atomic_uint`

```
typedef atomic<unsigned int> std::atomic_uint
```

`atomic_uint`

Definition at line 1012 of file `atomic`.

2.8.3.28 `atomic_uint16_t`

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

`atomic_uint16_t`

Definition at line 1054 of file `atomic`.

2.8.3.29 `atomic_uint32_t`

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

`atomic_uint32_t`

Definition at line 1060 of file `atomic`.

2.8.3.30 `atomic_uint64_t`

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

`atomic_uint64_t`

Definition at line 1066 of file `atomic`.

2.8.3.31 `atomic_uint8_t`

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

`atomic_uint8_t`

Definition at line 1048 of file `atomic`.

2.8.3.32 `atomic_uint_fast16_t`

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

`atomic_uint_fast16_t`

Definition at line 1104 of file `atomic`.

2.8.3.33 `atomic_uint_fast32_t`

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

`atomic_uint_fast32_t`

Definition at line 1110 of file `atomic`.

2.8.3.34 `atomic_uint_fast64_t`

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

`atomic_uint_fast64_t`

Definition at line 1116 of file `atomic`.

2.8.3.35 `atomic_uint_fast8_t`

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

`atomic_uint_fast8_t`

Definition at line 1098 of file `atomic`.

2.8.3.36 atomic_uint_least16_t

```
typedef atomic<uint_least16_t> std::atomic\_uint\_least16\_t
```

[atomic_uint_least16_t](#)

Definition at line 1079 of file atomic.

2.8.3.37 atomic_uint_least32_t

```
typedef atomic<uint_least32_t> std::atomic\_uint\_least32\_t
```

[atomic_uint_least32_t](#)

Definition at line 1085 of file atomic.

2.8.3.38 atomic_uint_least64_t

```
typedef atomic<uint_least64_t> std::atomic\_uint\_least64\_t
```

[atomic_uint_least64_t](#)

Definition at line 1091 of file atomic.

2.8.3.39 atomic_uint_least8_t

```
typedef atomic<uint_least8_t> std::atomic\_uint\_least8\_t
```

[atomic_uint_least8_t](#)

Definition at line 1073 of file atomic.

2.8.3.40 atomic_uintmax_t

```
typedef atomic<uintmax_t> std::atomic\_uintmax\_t
```

[atomic_uintmax_t](#)

Definition at line 1137 of file atomic.

2.8.3.41 atomic_uintptr_t

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

atomic_uintptr_t

Definition at line 1124 of file atomic.

2.8.3.42 atomic_ullong

```
typedef atomic<unsigned long long> std::atomic_ullong
```

atomic_ullong

Definition at line 1024 of file atomic.

2.8.3.43 atomic_ulong

```
typedef atomic<unsigned long> std::atomic_ulong
```

atomic_ulong

Definition at line 1018 of file atomic.

2.8.3.44 atomic_ushort

```
typedef atomic<unsigned short> std::atomic_ushort
```

atomic_ushort

Definition at line 1006 of file atomic.

2.8.3.45 atomic_wchar_t

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

atomic_wchar_t

Definition at line 1027 of file atomic.

2.8.3.46 memory_order

```
typedef enum std::memory_order std::memory_order
```

Enumeration for memory_order.

2.8.4 Enumeration Type Documentation

2.8.4.1 memory_order

```
enum std::memory_order
```

Enumeration for memory_order.

Definition at line 74 of file atomic_base.h.

2.8.5 Function Documentation

2.8.5.1 kill_dependency()

```
template<typename _Tp >  
_Tp std::kill_dependency (   
    _Tp __y ) [inline], [noexcept]
```

kill_dependency

Definition at line 131 of file atomic_base.h.

2.9 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct `std::__detail::Default_ranged_hash`
- struct `std::__detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys`
- struct `std::__detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Equality<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, Default_ranged_hash, false >`
- struct `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, Default_ranged_hash, true >`
- struct `std::__detail::Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >`
- struct `std::__detail::Hash_node<_Value, _Cache_hash_code >`
- struct `std::__detail::Hash_node<_Value, false >`
- struct `std::__detail::Hash_node<_Value, true >`
- struct `std::__detail::Hash_node_base`
- struct `std::__detail::Hash_node_value_base<_Value >`
- class `std::Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::Hashtable_alloc<_NodeAlloc >`
- struct `std::__detail::Hashtable_base<_Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`
- struct `std::__detail::Hashtable_ebo_helper<_Nm, _Tp, __use_ebo >`
- struct `std::__detail::Hashtable_ebo_helper<_Nm, _Tp, false >`
- struct `std::__detail::Hashtable_ebo_helper<_Nm, _Tp, true >`
- struct `std::__detail::Hashtable_traits<_Cache_hash_code, _Constant_iterators, _Unique_keys >`
- struct `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_itera`
- struct `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Insert_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`
- struct `std::__detail::Local_const_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`
- struct `std::__detail::Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_k`
- struct `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Mask_range_hashing`
- struct `std::__detail::Mod_range_hashing`

- struct `std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >`
- struct `std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >`
- struct `std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >`
- struct `std::__detail::_Power2_rehash_policy`
- struct `std::__detail::_Prime_rehash_policy`
- struct `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename... >`
- struct `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false_type >`
- struct `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true_type >`

Typedefs

- template<typename _Policy >
using `std::__detail::__has_load_factor` = typename _Policy::__has_load_factor
- template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >
using `std::__detail::__hash_code_for_local_iter` = _Hash_code_storage< `_Hash_code_base`< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > >

Functions

- `std::size_t std::__detail::__clp2` (std::size_t __n) noexcept
- template<class _Iterator >
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw` (_Iterator __first, _Iterator __last, `std::input_iterator_tag`)
- template<class _Iterator >
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw` (_Iterator __first, _Iterator __last, `std::forward_iterator_tag`)
- template<class _Iterator >
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw` (_Iterator __first, _Iterator __last)
- `__bucket_type * std::__detail::__Hashtable_alloc< _NodeAlloc >::M_allocate_buckets` (std::size_t __bkt_count)
- template<typename... _Args>
auto `std::__detail::__Hashtable_alloc< _NodeAlloc >::M_allocate_node` (_Args &&... __args) -> __node_type *
- void `std::__detail::__Hashtable_alloc< _NodeAlloc >::M_deallocate_buckets` (__bucket_type *, std::size_t __bkt_count)
- void `std::__detail::__Hashtable_alloc< _NodeAlloc >::M_deallocate_node` (__node_type * __n)
- void `std::__detail::__Hashtable_alloc< _NodeAlloc >::M_deallocate_node_ptr` (__node_type * __n)
- void `std::__detail::__Hashtable_alloc< _NodeAlloc >::M_deallocate_nodes` (__node_type * __n)
- bool `std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >::M_equal` (const `__hashtable` &) const
- bool `std::__detail::__Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >::M_equal` (const `__hashtable` &) const
- template<typename _InputIterator, typename _NodeGetter >
void `std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::M_insert_range` (_InputIterator __first, _InputIterator __last, const _NodeGetter &, `true_type`)
- template<typename _InputIterator, typename _NodeGetter >
void `std::__detail::__Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >::M_insert_range` (_InputIterator __first, _InputIterator __last, const _NodeGetter &, `false_type`)

- mapped_type & **std::__detail::Map_base**< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵
RehashPolicy, _Traits, true >::at (const key_type &__k)
- const mapped_type & **std::__detail::Map_base**< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash,
_RehashPolicy, _Traits, true >::at (const key_type &__k) const
- template<typename _Value, bool _Cache_hash_code>
bool **std::__detail::operator!=** (const [_Node_iterator_base](#)< _Value, _Cache_hash_code > &__x, const
[_Node_iterator_base](#)< _Value, _Cache_hash_code > &__y) noexcept
- template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
bool **std::__detail::operator!=** (const [_Local_iterator_base](#)< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵
__cache > &__x, const [_Local_iterator_base](#)< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- template<typename _Value, bool _Cache_hash_code>
bool **std::__detail::operator==** (const [_Node_iterator_base](#)< _Value, _Cache_hash_code > &__x, const
[_Node_iterator_base](#)< _Value, _Cache_hash_code > &__y) noexcept
- template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>
bool **std::__detail::operator==** (const [_Local_iterator_base](#)< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, _↵
__cache > &__x, const [_Local_iterator_base](#)< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)
- mapped_type & **std::__detail::Map_base**< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵
RehashPolicy, _Traits, true >::operator[] (const key_type &__k)
- mapped_type & **std::__detail::Map_base**< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _↵
RehashPolicy, _Traits, true >::operator[] (key_type &&__k)

2.9.1 Detailed Description

2.9.2 Function Documentation

2.9.2.1 __clp2()

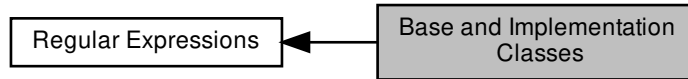
```
std::size_t std::__detail::__clp2 (
    std::size_t __n ) [inline], [noexcept]
```

Compute closest power of 2 not less than __n.

Definition at line 508 of file hashtable_policy.h.

2.10 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct `std::__detail::BracketMatcher<_TraitsT, __icase, __collate >`
- class `std::__detail::Compiler<_TraitsT >`
- class `std::__detail::Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`
- class `std::__detail::Scanner<_CharT >`
- class `std::__detail::StateSeq<_TraitsT >`

Typedefs

- template<typename _Iter, typename _TraitsT >
using `std::__detail::__disable_if_contiguous_iter` = `__enable_if_t< !__is_contiguous_iter< _Iter >::value, std::shared_ptr< const _NFA< _TraitsT > > >`
- template<typename _Iter, typename _TraitsT >
using `std::__detail::__enable_if_contiguous_iter` = `__enable_if_t< __is_contiguous_iter< _Iter >::value, std::shared_ptr< const _NFA< _TraitsT > > >`
- template<typename _CharT >
using `std::__detail::Matcher` = `std::function< bool(_CharT)>`
- typedef long `std::__detail::StateIdT`

Enumerations

- enum `std::__detail::Opcode` : int {
`_S_opcode_unknown, _S_opcode_alternative, _S_opcode_repeat, _S_opcode_backref,`
`_S_opcode_line_begin_assertion, _S_opcode_line_end_assertion, _S_opcode_word_boundary, _S_opcode_subexpr_lookahead,`
`_S_opcode_subexpr_begin, _S_opcode_subexpr_end, _S_opcode_dummy, _S_opcode_match,`
`_S_opcode_accept }`

Functions

- template<typename _TraitsT, typename _FwdIter >
`__enable_if_contiguous_iter< _FwdIter, _TraitsT > std::__detail::__compile_nfa` (`_FwdIter __first, _FwdIter __last, const typename _TraitsT::locale_type & __loc, regex_constants::syntax_option_type __flags`)

Variables

- static const _StateIdT **std::__detail::_S_invalid_state_id**

2.10.1 Detailed Description

2.10.2 Enumeration Type Documentation

2.10.2.1 _Opcode

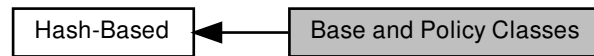
```
enum std::__detail::_Opcode : int
```

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

Definition at line 56 of file regex_automaton.h.

2.11 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



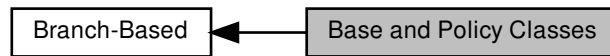
Classes

- [class `__gnu_pbds::detail::cc_ht_map`](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
- [class `__gnu_pbds::detail::gp_ht_map`](#)< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

2.11.1 Detailed Description

2.12 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



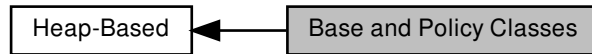
Classes

- [class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >`](#)
- [class `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)
- [class `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`](#)

2.12.1 Detailed Description

2.13 Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



Classes

- [class `__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >`](#)
- [class `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`](#)
- [class `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`](#)
- [class `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`](#)
- [class `__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >`](#)

2.13.1 Detailed Description

2.14 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution< _IntType >`
- class `std::geometric_distribution< _IntType >`
- class `std::negative_binomial_distribution< _IntType >`
- struct `std::negative_binomial_distribution< _IntType >::param_type`
- struct `std::geometric_distribution< _IntType >::param_type`
- struct `std::binomial_distribution< _IntType >::param_type`
- struct `std::bernoulli_distribution::param_type`

Functions

- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >
bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`
- `template<typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)`

2.14.1 Detailed Description

2.14.2 Function Documentation

2.14.2.1 `operator!=()` [1/4]

```
bool std::operator!=(  
    const std::bernoulli\_distribution & __d1,  
    const std::bernoulli\_distribution & __d2 ) [inline]
```

Return true if two Bernoulli distributions have different parameters.

Definition at line 3690 of file random.h.

2.14.2.2 `operator!=()` [2/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::binomial\_distribution< _IntType > & __d1,  
    const std::binomial\_distribution< _IntType > & __d2 ) [inline]
```

Return true if two binomial distributions are different.

Definition at line 3965 of file random.h.

2.14.2.3 `operator!=()` [3/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::geometric\_distribution< _IntType > & __d1,  
    const std::geometric\_distribution< _IntType > & __d2 ) [inline]
```

Return true if two geometric distributions have different parameters.

Definition at line 4144 of file random.h.

2.14.2.4 `operator!=()` [4/4]

```
template<typename _IntType >  
bool std::operator!=(  
    const std::negative\_binomial\_distribution< _IntType > & __d1,  
    const std::negative\_binomial\_distribution< _IntType > & __d2 ) [inline]
```

Return true if two negative binomial distributions are different.

Definition at line 4398 of file random.h.

2.14.2.5 operator<<() [1/2]

```
template<typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::bernoulli_distribution & __x )
```

Inserts a `bernoulli_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>bernoulli_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 986 of file `bits/random.tcc`.

References `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, and `std::ios_base::precision()`.

2.14.2.6 `operator<<()` [2/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::geometric_distribution< _IntType > & __x )
```

Inserts a `geometric_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>geometric_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1067 of file `bits/random.tcc`.

References `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, and `std::ios_base::precision()`.

2.14.2.7 `operator>>()` [1/2]

```
template<typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::bernoulli_distribution & __x ) [inline]
```

Extracts a `bernoulli_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

\leftrightarrow __is	An input stream.
\leftrightarrow __x	A bernoulli_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

Definition at line 3720 of file random.h.

References std::bernoulli_distribution::param().

2.14.2.8 operator>>() [2/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::geometric_distribution< _IntType > & __x )
```

Extracts a geometric_distribution random number distribution __x from the input stream __is.

Parameters

\leftrightarrow __is	An input stream.
\leftrightarrow __x	A geometric_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

Definition at line 1090 of file bits/random.tcc.

References std::ios_base::flags(), std::geometric_distribution< _IntType >::param(), and std::skipws().

2.15 Binary Search

Collaboration diagram for Binary Search:



Functions

- `template<typename _ForwardIterator, typename _Tp >`
`constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`
`constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`

2.15.1 Detailed Description

These algorithms are variations of a classic binary search, and all assume that the sequence being searched is already sorted.

The number of comparisons will be logarithmic (and as few as possible). The number of steps through the sequence will be logarithmic for random-access iterators (e.g., pointers), and linear otherwise.

The LWG has passed Defect Report 270, which notes: *The proposed resolution reinterprets binary search. Instead of thinking about searching for a value in a sorted range, we view that as an important special case of a more general algorithm: searching for the partition point in a partitioned range. We also add a guarantee that the old wording did not: we ensure that the upper bound is no earlier than the lower bound, that the pair returned by equal_range is a valid range, and that the first part of that pair is the lower bound.*

The actual effect of the first sentence is that a comparison functor passed by the user doesn't necessarily need to induce a strict weak ordering relation. Rather, it partitions the range.

2.15.2 Function Documentation

2.15.2.1 `binary_search()` [1/2]

```
template<typename _ForwardIterator , typename _Tp >
constexpr bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val )
```

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

Definition at line 2267 of file `stl_algo.h`.

2.15.2.2 `binary_search()` [2/2]

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >
constexpr bool std::binary_search (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp )
```

Determines whether an element exists in a range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

True if `__val` (or its equivalent) is in `[__first,__last]`.

Note that this does not actually return an iterator to `__val`. For that, use `std::find` or a container's specialized find member functions.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2301 of file `stl_algo.h`.

2.15.2.3 `equal_range()` [1/2]

```
template<typename _ForwardIterator , typename _Tp >
constexpr pair<_ForwardIterator, _ForwardIterator> std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val ) [inline]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val),
               upper_bound(__first, __last, __val))
```

but does not actually call those functions.

Definition at line 2196 of file `stl_algo.h`.

2.15.2.4 `equal_range()` [2/2]

```
template<typename _ForwardIterator , typename _Tp , typename _Compare >
constexpr pair<_ForwardIterator, _ForwardIterator> std::equal_range (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp ) [inline]
```

Finds the largest subrange in which `__val` could be inserted at any place in it without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An pair of iterators defining the subrange.

This is equivalent to

```
std::make_pair(lower_bound(__first, __last, __val, __comp),
               upper_bound(__first, __last, __val, __comp))
```

but does not actually call those functions.

Definition at line 2233 of file `stl_algo.h`.

2.15.2.5 `lower_bound()` [1/2]

```
template<typename _ForwardIterator, typename _Tp>
constexpr _ForwardIterator std::lower_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val ) [inline]
```

Finds the first position in which *val* could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element *not less than val*, or `end()` if every element is less than *val*.

Definition at line 1348 of file `stl_algo.h`.

2.15.2.6 `lower_bound()` [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare>
constexpr _ForwardIterator std::lower_bound (
```

```

_FowardIterator __first,
_FowardIterator __last,
const _Tp & __val,
_Compare __comp ) [inline]

```

Finds the first position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element *not less than* `__val`, or `end()` if every element is less than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2036 of file `stl_algo.h`.

2.15.2.7 `upper_bound()` [1/2]

```

template<typename _ForwardIterator , typename _Tp >
constexpr _ForwardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val ) [inline]

```

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

Definition at line 2092 of file `stl_algo.h`.

2.15.2.8 `upper_bound()` [2/2]

```
template<typename _ForwardIterator, typename _Tp, typename _Compare >
constexpr _ForwardIterator std::upper_bound (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __val,
    _Compare __comp ) [inline]
```

Finds the last position in which `__val` could be inserted without changing the ordering.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__val</code>	The search term.
<code>__comp</code>	A functor to use for comparisons.

Returns

An iterator pointing to the first element greater than `__val`, or `end()` if no elements are greater than `__val`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2123 of file `stl_algo.h`.

2.16 Binder Classes

Collaboration diagram for Binder Classes:



Namespaces

- [std::placeholders](#)

Classes

- struct [std::_Placeholder<_Num>](#)
- class [std::binder1st<_Operation>](#)
- class [std::binder2nd<_Operation>](#)
- struct [std::is_bind_expression<_Tp>](#)
- struct [std::is_bind_expression<_Bind<_Signature>>](#)
- struct [std::is_bind_expression<_Bind_result<_Result, _Signature>>](#)
- struct [std::is_bind_expression<const _Bind<_Signature>>](#)
- struct [std::is_bind_expression<const _Bind_result<_Result, _Signature>>](#)
- struct [std::is_bind_expression<const volatile _Bind<_Signature>>](#)
- struct [std::is_bind_expression<const volatile _Bind_result<_Result, _Signature>>](#)
- struct [std::is_bind_expression<volatile _Bind<_Signature>>](#)
- struct [std::is_bind_expression<volatile _Bind_result<_Result, _Signature>>](#)
- struct [std::is_placeholder<_Tp>](#)
- struct [std::is_placeholder<_Placeholder<_Num>>](#)

Functions

- template<typename _Func, typename... _BoundArgs>
constexpr [_Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type](#) [std::bind](#) (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Result, typename _Func, typename... _BoundArgs>
constexpr [_Bindres_helper<_Result, _Func, _BoundArgs...>::type](#) [std::bind](#) (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Operation, typename _Tp>
[binder1st<_Operation>](#) [std::bind1st](#) (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
[binder2nd<_Operation>](#) [std::bind2nd](#) (const _Operation &__fn, const _Tp &__x)

2.16.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, `B's operator()` is called with a single argument `y`. The return value is the value of `f(x, y)`. `B` can be *called* with various arguments (`y1, y2, ...`) and will in turn call `f(x, y1), f(x, y2), ...`

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd(std::minus<float>(), 1.3)` will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform `1.3 - x` instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(), 5)`.

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::placeholders::_1, x)`.

2.16.2 Function Documentation

2.16.2.1 `bind()` [1/2]

```
template<typename _Func , typename... _BoundArgs>
constexpr _Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs...>::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args ) [inline]
```

Function template for `std::bind`.

Definition at line 785 of file `functional`.

2.16.2.2 `bind()` [2/2]

```
template<typename _Result , typename _Func , typename... _BoundArgs>
constexpr _Bindres_helper<_Result, _Func, _BoundArgs...>::type std::bind (
    _Func && __f,
    _BoundArgs &&... __args ) [inline]
```

Function template for `std::bind<R>`.

Definition at line 809 of file `functional`.

2.16.2.3 bind1st()

```
template<typename _Operation , typename _Tp >  
binder1st<_Operation> std::bind1st (   
    const _Operation & __fn,  
    const _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 135 of file binders.h.

2.16.2.4 bind2nd()

```
template<typename _Operation , typename _Tp >  
binder2nd<_Operation> std::bind2nd (   
    const _Operation & __fn,  
    const _Tp & __x ) [inline]
```

One of the [binder functors](#).

Definition at line 170 of file binders.h.

2.17 Bit manipulation

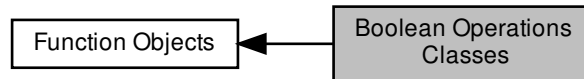
Collaboration diagram for Bit manipulation:



Utilities for examining and manipulating individual bits.

2.18 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

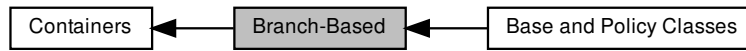
- struct `std::logical_and< _Tp >`
- struct `std::logical_and< void >`
- struct `std::logical_not< _Tp >`
- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`

2.18.1 Detailed Description

Here are wrapper functors for Boolean operations: `&&`, `||`, and `!`.

2.19 Branch-Based

Collaboration diagram for Branch-Based:



Modules

- [Base and Policy Classes](#)

Classes

- [class `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >`](#)
- [class `__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >`](#)
- [class `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`](#)

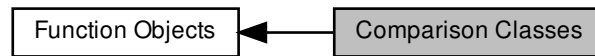
Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

2.19.1 Detailed Description

2.20 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct `std::equal_to< _Tp >`
- struct `std::equal_to< void >`
- struct `std::greater< _Tp >`
- struct `std::greater< void >`
- struct `std::greater_equal< _Tp >`
- struct `std::greater_equal< void >`
- struct `std::less< _Tp >`
- struct `std::less< void >`
- struct `std::less_equal< _Tp >`
- struct `std::less_equal< void >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< void >`

2.20.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like `<`.

2.21 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- struct `std::complex< _Tp >`
- struct `std::complex< double >`
- struct `std::complex< float >`
- struct `std::complex< long double >`

Functions

- constexpr `std::complex< float >::complex` (const `complex< double >` &)
- constexpr `std::complex< float >::complex` (const `complex< long double >` &)
- constexpr `std::complex< double >::complex` (const `complex< long double >` &)
- template<typename `_Tp` >
`_Tp std::__complex_abs` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`_Tp std::__complex_arg` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_cos` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_cosh` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_exp` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_log` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_pow` (const `complex< _Tp >` &__x, const `complex< _Tp >` &__y)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_pow_unsigned` (`complex< _Tp >` __x, unsigned __n)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_sin` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_sinh` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_sqrt` (const `complex< _Tp >` &__z)
- template<typename `_Tp` >
`complex< _Tp > std::__complex_tan` (const `complex< _Tp >` &__z)

- `template<typename _Tp >`
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`
`_Tp constexpr std::norm (const complex< _Tp > &)`
- `constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)`
- `template<typename _Up >`
`constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Up >`
`constexpr complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Up >`
`constexpr complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)`
- `constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &)`
- `template<typename _Up >`
`constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `constexpr complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &)`
- `template<typename _Up >`
`constexpr complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &)`
- `template<typename _Tp, typename _CharT, class _Traits >`
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`

- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar` (const _Tp &__rho,
const _Up &__theta)
- `template<typename _Tp >`
`complex< _Tp > std::pow` (const `complex< _Tp >` &, int)
- `template<typename _Tp >`
`complex< _Tp > std::pow` (const `complex< _Tp >` &, const _Tp &)
- `template<typename _Tp >`
`complex< _Tp > std::pow` (const `complex< _Tp >` &, const `complex< _Tp >` &)
- `template<typename _Tp >`
`complex< _Tp > std::pow` (const _Tp &, const `complex< _Tp >` &)
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const `std::complex<`
`_Tp >` &__x, const _Up &__y)
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const _Tp &__x,
const `std::complex< _Tp >` &__y)
- `template<typename _Tp, typename _Up >`
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow` (const `std::complex<`
`_Tp >` &__x, const `std::complex< _Up >` &__y)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &__x, const _Tp &__y)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow` (const _Tp &__x, const `std::complex< _Tp >` &__y)
- `template<typename _Tp >`
`std::complex< _Tp > std::tr1::pow` (const `std::complex< _Tp >` &__x, const `std::complex< _Tp >` &__y)
- `template<typename _Tp >`
`constexpr _Tp std::real` (const `complex< _Tp >` &__z)
- `template<typename _Tp >`
`complex< _Tp > std::sin` (const `complex< _Tp >` &)
- `template<typename _Tp >`
`complex< _Tp > std::sinh` (const `complex< _Tp >` &)
- `template<typename _Tp >`
`complex< _Tp > std::sqrt` (const `complex< _Tp >` &)
- `template<typename _Tp >`
`complex< _Tp > std::tan` (const `complex< _Tp >` &)
- `template<typename _Tp >`
`complex< _Tp > std::tanh` (const `complex< _Tp >` &)

- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator+` (const `complex< _Tp >` &__x, const `complex< _Tp >` &__y)
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator+` (const `complex< _Tp >` &__x, const _Tp &__y)
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator+` (const _Tp &__x, const `complex< _Tp >` &__y)

- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator-` (const `complex< _Tp >` &__x, const `complex< _Tp >` &__y)
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator-` (const `complex< _Tp >` &__x, const _Tp &__y)
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator-` (const _Tp &__x, const `complex< _Tp >` &__y)

- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`

- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

2.21.1 Detailed Description

Classes and functions for complex numbers.

2.21.2 Function Documentation

2.21.2.1 `abs()`

```
template<typename _Tp >
_Tp std::abs (
    const complex< _Tp > & __z ) [inline]
```

Return magnitude of `z`.

Definition at line 629 of file `complex`.

Referenced by `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

2.21.2.2 `arg()`

```
template<typename _Tp >
_Tp std::arg (
    const complex< _Tp > & __z ) [inline]
```

Return phase angle of z .

Definition at line 656 of file `complex`.

2.21.2.3 `conj()`

```
template<typename _Tp >
constexpr complex< _Tp > std::conj (
    const complex< _Tp > & __z ) [inline]
```

Return complex conjugate of z .

Definition at line 708 of file `complex`.

2.21.2.4 `cos()`

```
template<typename _Tp >
complex< _Tp > std::cos (
    const complex< _Tp > & __z ) [inline]
```

Return complex cosine of z .

Definition at line 740 of file `complex`.

2.21.2.5 `cosh()`

```
template<typename _Tp >
complex< _Tp > std::cosh (
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic cosine of z .

Definition at line 770 of file `complex`.

2.21.2.6 exp()

```
template<typename _Tp >
complex< _Tp > std::exp (
    const complex< _Tp > & __z ) [inline]
```

Return complex base e exponential of z.

Definition at line 796 of file complex.

2.21.2.7 fabs()

```
template<typename _Tp >
std::complex< _Tp > std::tr1::fabs (
    const std::complex< _Tp > & __z ) [inline]
```

fabs(__z) [8.1.8].

Definition at line 309 of file tr1/complex.

2.21.2.8 log()

```
template<typename _Tp >
complex< _Tp > std::log (
    const complex< _Tp > & __z ) [inline]
```

Return complex natural logarithm of z.

Definition at line 823 of file complex.

Referenced by `std::generate_canonical()`, `std::normal_distribution< result_type >::operator()()`, `std::gamma_distribution< result_type >::operator()()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

2.21.2.9 log10()

```
template<typename _Tp >
complex< _Tp > std::log10 (
    const complex< _Tp > & __z ) [inline]
```

Return complex base 10 logarithm of z.

Definition at line 828 of file complex.

2.21.2.10 norm()

```
template<typename _Tp >
_Tp constexpr std::norm (
    const complex< _Tp > & ) [inline]
```

Return z magnitude squared.

Definition at line 692 of file complex.

2.21.2.11 operator!=() [1/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return false if x is equal to y .

Definition at line 482 of file complex.

2.21.2.12 operator!=() [2/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return false if x is equal to y .

Definition at line 487 of file complex.

2.21.2.13 operator!=() [3/3]

```
template<typename _Tp >
constexpr bool std::operator!= (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return false if x is equal to y .

Definition at line 492 of file complex.

2.21.2.14 operator*() [1/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator* (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x times y .

Definition at line 391 of file complex.

2.21.2.15 operator*() [2/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator* (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x times y .

Definition at line 400 of file complex.

2.21.2.16 operator*() [3/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator* (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x times y .

Definition at line 409 of file complex.

2.21.2.17 operator*=() [1/2]

```
template<typename _Tp >
constexpr complex< _Tp > & std::complex< _Tp >::operator*= (
    const _Tp & __t )
```

Multiply this complex number by a scalar.

Definition at line 250 of file complex.

2.21.2.18 operator*=() [2/2]

```
template<typename _Tp >
template<typename _Up >
constexpr complex< _Tp > & std::complex< _Tp >::operator*= (
    const complex< _Up > & __z )
```

Multiply this complex number by another.

Definition at line 304 of file complex.

2.21.2.19 operator+() [1/4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator+ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 331 of file complex.

2.21.2.20 operator+() [2/4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator+ (
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 340 of file complex.

2.21.2.21 operator+() [3/4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator+ (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x plus y.

Definition at line 349 of file complex.

2.21.2.22 operator+() [4 / 4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator+ (
    const complex<_Tp > & __x ) [inline]
```

Return x.

Definition at line 450 of file complex.

2.21.2.23 operator+=()

```
template<typename _Tp >
template<typename _Up >
constexpr complex<_Tp > & std::complex<_Tp >::operator+= (
    const complex<_Up > & __z )
```

Add another complex number to this one.

Definition at line 281 of file complex.

2.21.2.24 operator-() [1 / 4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator- (
    const complex<_Tp > & __x,
    const complex<_Tp > & __y ) [inline]
```

Return new complex value x minus y.

Definition at line 361 of file complex.

2.21.2.25 operator-() [2 / 4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator- (
    const complex<_Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x minus y.

Definition at line 370 of file complex.

2.21.2.26 operator-() [3/4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator- (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x minus y .

Definition at line 379 of file complex.

2.21.2.27 operator-() [4/4]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator- (
    const complex< _Tp > & __x ) [inline]
```

Return complex negation of x .

Definition at line 456 of file complex.

2.21.2.28 operator-=()

```
template<typename _Tp >
template<typename _Up >
constexpr complex< _Tp > & std::complex< _Tp >::operator-= (
    const complex< _Up > & __z )
```

Subtract another complex number from this one.

Definition at line 292 of file complex.

2.21.2.29 operator/() [1/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator/ (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return new complex value x divided by y .

Definition at line 421 of file complex.

2.21.2.30 operator/() [2/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator/ (
    const complex<_Tp > & __x,
    const _Tp & __y ) [inline]
```

Return new complex value x divided by y.

Definition at line 430 of file complex.

2.21.2.31 operator/() [3/3]

```
template<typename _Tp >
constexpr complex<_Tp> std::operator/ (
    const _Tp & __x,
    const complex<_Tp > & __y ) [inline]
```

Return new complex value x divided by y.

Definition at line 439 of file complex.

2.21.2.32 operator/=() [1/2]

```
template<typename _Tp >
constexpr complex<_Tp > & std::complex<_Tp >::operator/= (
    const _Tp & __t )
```

Divide this complex number by a scalar.

Definition at line 260 of file complex.

2.21.2.33 operator/=() [2/2]

```
template<typename _Tp >
template<typename _Up >
constexpr complex<_Tp > & std::complex<_Tp >::operator/= (
    const complex<_Up > & __z )
```

Divide this complex number by another.

Definition at line 317 of file complex.

2.21.2.34 operator<<()

```
template<typename _Tp , typename _CharT , class _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
    basic_ostream< _CharT, _Traits > & __os,
    const complex< _Tp > & __x )
```

Insertion operator for complex values.

Definition at line 554 of file complex.

2.21.2.35 operator=() [1/2]

```
template<typename _Tp >
constexpr complex< _Tp > & std::complex< _Tp >::operator= (
    const _Tp & __t )
```

Assign a scalar to this complex number.

Definition at line 240 of file complex.

2.21.2.36 operator=() [2/2]

```
template<typename _Tp >
template<typename _Up >
constexpr complex< _Tp > & std::complex< _Tp >::operator= (
    const complex< _Up > & __z )
```

Assign another complex number to this one.

Definition at line 270 of file complex.

2.21.2.37 operator==() [1/3]

```
template<typename _Tp >
constexpr bool std::operator==(
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return true if x is equal to y.

Definition at line 463 of file complex.

2.21.2.38 operator==() [2/3]

```
template<typename _Tp >
constexpr bool std::operator==(
    const complex< _Tp > & __x,
    const _Tp & __y ) [inline]
```

Return true if *x* is equal to *y*.

Definition at line 468 of file `complex`.

2.21.2.39 operator==() [3/3]

```
template<typename _Tp >
constexpr bool std::operator==(
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return true if *x* is equal to *y*.

Definition at line 474 of file `complex`.

2.21.2.40 operator>>()

```
template<typename _Tp , typename _CharT , class _Traits >
basic_istream<_CharT, _Traits>& std::operator>> (
    basic_istream< _CharT, _Traits > & __is,
    complex< _Tp > & __x )
```

Extraction operator for complex values.

Definition at line 500 of file `complex`.

2.21.2.41 polar()

```
template<typename _Tp >
complex< _Tp > std::polar (
    const _Tp & __rho,
    const _Tp & __theta = 0 ) [inline]
```

Return complex with magnitude *rho* and angle *theta*.

Definition at line 700 of file `complex`.

2.21.2.42 pow() [1/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __z,
    int __n ) [inline]
```

Return x to the y 'th power.

Definition at line 1018 of file complex.

Referenced by `std::gamma_distribution< result_type >::operator()()`.

2.21.2.43 pow() [2/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const _Tp & __y )
```

Return x to the y 'th power.

Definition at line 1027 of file complex.

2.21.2.44 pow() [3/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const complex< _Tp > & __x,
    const complex< _Tp > & __y ) [inline]
```

Return x to the y 'th power.

Definition at line 1066 of file complex.

2.21.2.45 pow() [4/5]

```
template<typename _Tp >
complex< _Tp > std::pow (
    const _Tp & __x,
    const complex< _Tp > & __y ) [inline]
```

Return x to the y 'th power.

Definition at line 1072 of file complex.

2.21.2.46 pow() [5/5]

```
template<typename _Tp , typename _Up >
std::complex<typename __gnu_cxx::__promote_2<_Tp, _Up>::__type> std::tr1::pow (
    const std::complex< _Tp > & __x,
    const _Up & __y ) [inline]
```

Additional overloads [8.1.9].

Definition at line 350 of file tr1/complex.

2.21.2.47 sin()

```
template<typename _Tp >
complex< _Tp > std::sin (
    const complex< _Tp > & __z ) [inline]
```

Return complex sine of z.

Definition at line 858 of file complex.

2.21.2.48 sinh()

```
template<typename _Tp >
complex< _Tp > std::sinh (
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic sine of z.

Definition at line 888 of file complex.

2.21.2.49 sqrt()

```
template<typename _Tp >
complex< _Tp > std::sqrt (
    const complex< _Tp > & __z ) [inline]
```

Return complex square root of z.

Definition at line 932 of file complex.

Referenced by `std::normal_distribution< result_type >::operator()()`, and `std::student_t_distribution< _RealType >::operator()()`.

2.21.2.50 tan()

```
template<typename _Tp >  
complex< _Tp > std::tan (  
    const complex< _Tp > & __z ) [inline]
```

Return complex tangent of z.

Definition at line 959 of file complex.

2.21.2.51 tanh()

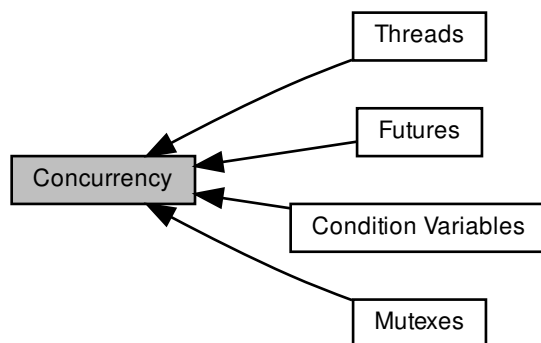
```
template<typename _Tp >  
complex< _Tp > std::tanh (  
    const complex< _Tp > & __z ) [inline]
```

Return complex hyperbolic tangent of z.

Definition at line 987 of file complex.

2.22 Concurrency

Collaboration diagram for Concurrency:



Modules

- [Condition Variables](#)
- [Futures](#)
- [Mutexes](#)
- [Threads](#)

2.22.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

2.23 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class `std::condition_variable`
- class `std::_V2::condition_variable_any`

Enumerations

- enum `std::cv_status` { `no_timeout`, `timeout` }

Functions

- void `std::notify_all_at_thread_exit` (`condition_variable` &, `unique_lock`< `mutex` >)

2.23.1 Detailed Description

Classes for `condition_variable` support.

2.23.2 Enumeration Type Documentation

2.23.2.1 `cv_status`

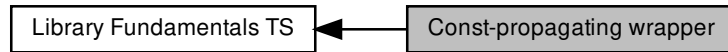
```
enum std::cv_status [strong]
```

`cv_status`

Definition at line 68 of file `condition_variable`.

2.24 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



Classes

- class `std::experimental::fundamentals_v2::propagate_const< _Tp >`

Functions

- `template<typename _Tp >`
`constexpr const _Tp & std::experimental::fundamentals_v2::get_underlying (const propagate_const< _Tp > &__pt) noexcept`
- `template<typename _Tp >`
`constexpr _Tp & std::experimental::fundamentals_v2::get_underlying (propagate_const< _Tp > &__pt) noexcept`
- `template<typename _Tp >`
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, nullptr_t)`
- `template<typename _Tp >`
`constexpr bool std::experimental::fundamentals_v2::operator!= (nullptr_t, const propagate_const< _Tp > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator!= (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator!= (const _Tp &__t, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator< (const _Tp &__t, const propagate_const< _Up > &__pu)`

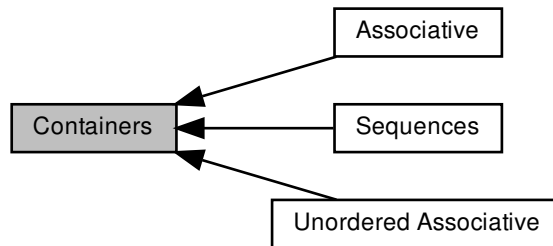
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator<= (const _Tp &__t, const propagate_const<`
`_Up > &__pu)`
- `template<typename _Tp >`
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`
`nullptr_t)`
- `template<typename _Tp >`
`constexpr bool std::experimental::fundamentals_v2::operator== (nullptr_t, const propagate_const< _Tp >`
`&__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator== (const _Tp &__t, const propagate_const<`
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const< _Tp > &__pt, const`
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const< _Tp > &__pt, const`
`_Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator> (const _Tp &__t, const propagate_const< ↵`
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const< _Tp > &__pt,`
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const< _Tp > &__pt,`
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`
`constexpr bool std::experimental::fundamentals_v2::operator>= (const _Tp &__t, const propagate_const<`
`_Up > &__pu)`
- `template<typename _Tp >`
`constexpr void std::experimental::fundamentals_v2::swap (propagate_const< _Tp > &__pt, propagate_const<`
`_Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

2.24.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".

2.25 Containers

Collaboration diagram for Containers:



Modules

- [Associative](#)
- [Sequences](#)
- [Unordered Associative](#)

2.25.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

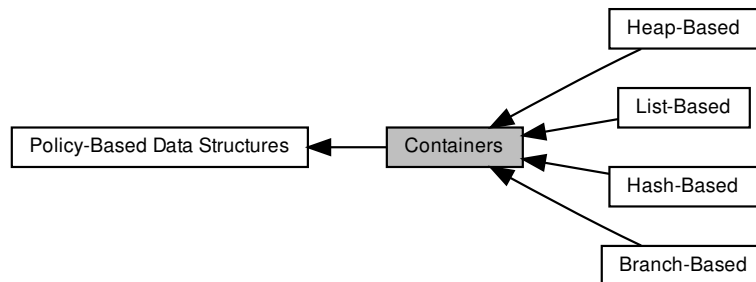
Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into [Sequences](#) and [Associative Containers](#). [Unordered Associative Containers](#).

2.26 Containers

Collaboration diagram for Containers:



Modules

- [Branch-Based](#)
- [Hash-Based](#)
- [Heap-Based](#)
- [List-Based](#)

2.26.1 Detailed Description

2.27 Data Structure Type

Collaboration diagram for Data Structure Type:



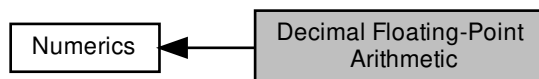
Classes

- struct [__gnu_pbds::associative_tag](#)
- struct [__gnu_pbds::basic_branch_tag](#)
- struct [__gnu_pbds::basic_hash_tag](#)
- struct [__gnu_pbds::binary_heap_tag](#)
- struct [__gnu_pbds::binomial_heap_tag](#)
- struct [__gnu_pbds::cc_hash_tag](#)
- struct [__gnu_pbds::container_tag](#)
- struct [__gnu_pbds::gp_hash_tag](#)
- struct [__gnu_pbds::list_update_tag](#)
- struct [__gnu_pbds::ov_tree_tag](#)
- struct [__gnu_pbds::pairing_heap_tag](#)
- struct [__gnu_pbds::pat_trie_tag](#)
- struct [__gnu_pbds::priority_queue_tag](#)
- struct [__gnu_pbds::rb_tree_tag](#)
- struct [__gnu_pbds::rc_binomial_heap_tag](#)
- struct [__gnu_pbds::sequence_tag](#)
- struct [__gnu_pbds::splay_tree_tag](#)
- struct [__gnu_pbds::string_tag](#)
- struct [__gnu_pbds::thin_heap_tag](#)
- struct [__gnu_pbds::tree_tag](#)
- struct [__gnu_pbds::trie_tag](#)

2.27.1 Detailed Description

2.28 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

- [std::decimal](#)

Classes

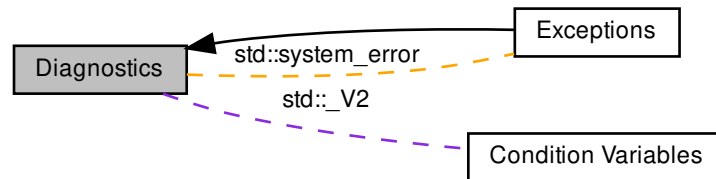
- class [std::decimal::decimal128](#)
- class [std::decimal::decimal32](#)
- class [std::decimal::decimal64](#)

2.28.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

2.29 Diagnostics

Collaboration diagram for Diagnostics:



Modules

- [Exceptions](#)

Classes

- class [std::_V2::error_category](#)
- struct [std::error_code](#)
- struct [std::error_condition](#)
- struct [std::hash< error_code >](#)
- struct [std::is_error_code_enum< _Tp >](#)
- struct [std::is_error_condition_enum< _Tp >](#)
- class [std::system_error](#)

Functions

- **std::_V2::error_category::error_category** (const [error_category](#) &)=delete
- **std::error_code::error_code** (int __v, const error_category &__cat) noexcept
- template<typename _ErrorCodeEnum , typename = typename enable_if<is_error_code_enum<_ErrorCodeEnum>::value::type>
std::error_code::error_code (_ErrorCodeEnum __e) noexcept
- **std::error_condition::error_condition** (int __v, const error_category &__cat) noexcept
- template<typename _ErrorConditionEnum , typename = typename enable_if<is_error_condition_enum<_ErrorConditionEnum>::value::type>
std::error_condition::error_condition (_ErrorConditionEnum __e) noexcept
- **std::system_error::system_error** ([error_code](#) __ec=[error_code](#)())
- **std::system_error::system_error** ([error_code](#) __ec, const [string](#) &__what)
- **std::system_error::system_error** ([error_code](#) __ec, const char * __what)
- **std::system_error::system_error** (int __v, const error_category &__ecat, const char * __what)
- **std::system_error::system_error** (int __v, const error_category &__ecat)
- **std::system_error::system_error** (int __v, const error_category &__ecat, const [string](#) &__what)
- **std::system_error::system_error** (const [system_error](#) &)=default

- void **std::error_code::assign** (int __v, const error_category &__cat) noexcept
- void **std::error_condition::assign** (int __v, const error_category &__cat) noexcept
- const error_category & **std::error_code::category** () const noexcept
- const error_category & **std::error_condition::category** () const noexcept
- void **std::error_code::clear** () noexcept
- void **std::error_condition::clear** () noexcept
- const [error_code](#) & **std::system_error::code** () const noexcept
- virtual [error_condition](#) **std::_V2::error_category::default_error_condition** (int __i) const noexcept
- [error_condition](#) **std::error_code::default_error_condition** () const noexcept
- virtual bool **std::_V2::error_category::equivalent** (int __i, const [error_condition](#) &__cond) const noexcept
- virtual bool **std::_V2::error_category::equivalent** (const [error_code](#) &__code, int __i) const noexcept
- const [error_category](#) & **std::_V2::generic_category** () noexcept
- [error_condition](#) **make_error_condition** (errc __e) noexcept
- virtual [string](#) **std::_V2::error_category::message** (int) const =0
- _GLIBCXX_DEFAULT_ABI_TAG [string](#) **std::error_code::message** () const
- _GLIBCXX_DEFAULT_ABI_TAG [string](#) **std::error_condition::message** () const
- virtual const char * **std::_V2::error_category::name** () const noexcept=0
- **std::error_code::operator bool** () const noexcept
- **std::error_condition::operator bool** () const noexcept
- bool **std::_V2::error_category::operator!=** (const [error_category](#) &__other) const noexcept
- bool **operator!=** (const [error_code](#) &__lhs, const [error_code](#) &__rhs) noexcept
- bool **operator!=** (const [error_code](#) &__lhs, const [error_condition](#) &__rhs) noexcept
- bool **operator!=** (const [error_condition](#) &__lhs, const [error_code](#) &__rhs) noexcept
- bool **operator!=** (const [error_condition](#) &__lhs, const [error_condition](#) &__rhs) noexcept
- size_t **std::hash< error_code >::operator()** (const [error_code](#) &__e) const noexcept
- bool **std::_V2::error_category::operator<** (const [error_category](#) &__other) const noexcept
- bool **operator<** (const [error_condition](#) &__lhs, const [error_condition](#) &__rhs) noexcept
- [error_category](#) & **std::_V2::error_category::operator=** (const [error_category](#) &)=delete
- template<typename _ErrorCodeEnum >
[enable_if< is_error_code_enum< _ErrorCodeEnum >::value, \[error_code\]\(#\) & >::type](#) **std::error_code::operator=** (_ErrorCodeEnum __e) noexcept
- template<typename _ErrorConditionEnum >
[enable_if< is_error_condition_enum< _ErrorConditionEnum >::value, \[error_condition\]\(#\) & >::type](#) **std::error_condition::operator=** (_ErrorConditionEnum __e) noexcept
- [system_error](#) & **std::system_error::operator=** (const [system_error](#) &)=default
- bool **std::_V2::error_category::operator==** (const [error_category](#) &__other) const noexcept
- bool **operator==** (const [error_code](#) &__lhs, const [error_code](#) &__rhs) noexcept
- bool **operator==** (const [error_code](#) &__lhs, const [error_condition](#) &__rhs) noexcept
- bool **operator==** (const [error_condition](#) &__lhs, const [error_condition](#) &__rhs) noexcept
- bool **operator==** (const [error_condition](#) &__lhs, const [error_code](#) &__rhs) noexcept
- const [error_category](#) & **std::_V2::system_category** () noexcept
- int **std::error_code::value** () const noexcept
- int **std::error_condition::value** () const noexcept

Variables

- [error_code](#) **std::make_error_code** (errc) noexcept
- [error_condition](#) **make_error_condition** (errc) noexcept
- [error_code](#) **make_error_code** (errc __e) noexcept

2.29.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

2.29.2 Function Documentation

2.29.2.1 `generic_category()`

```
const error\_category& std::_V2::generic_category ( ) [noexcept]
```

Error category for `errno` error codes.

2.29.2.2 `make_error_code()`

```
error\_code make_error_code (
    errc __e ) [related]
```

Definition at line 239 of file `system_error`.

2.29.2.3 `make_error_condition()`

```
error\_condition make_error_condition (
    errc __e ) [related]
```

Create an `error_condition` representing a standard `errc` condition.

Definition at line 335 of file `system_error`.

2.29.2.4 `operator!=()` [1/2]

```
bool operator!= (
    const error\_code & __lhs,
    const error\_code & __rhs ) [related]
```

Definition at line 398 of file `system_error`.

2.29.2.5 operator!=() [2/2]

```
bool operator!=(  
    const error_condition & __lhs,  
    const error_condition & __rhs ) [related]
```

Definition at line 415 of file system_error.

2.29.2.6 operator<()

```
bool operator<(  
    const error_condition & __lhs,  
    const error_condition & __rhs ) [related]
```

Define an ordering for error_condition objects.

Definition at line 379 of file system_error.

2.29.2.7 operator==() [1/2]

```
bool operator==(  
    const error_code & __lhs,  
    const error_code & __rhs ) [related]
```

Definition at line 342 of file system_error.

2.29.2.8 operator==() [2/2]

```
bool operator==(  
    const error_condition & __lhs,  
    const error_condition & __rhs ) [related]
```

Definition at line 357 of file system_error.

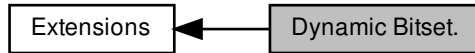
2.29.2.9 system_category()

```
const error_category& std::_V2::system_category ( ) [noexcept]
```

Error category for other error codes defined by the OS.

2.30 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



Classes

- struct `std::tr2::__dynamic_bitset_base< _WordT, _Alloc >`
- class `std::tr2::dynamic_bitset< _WordT, _Alloc >`
- class `std::tr2::dynamic_bitset< _WordT, _Alloc >::reference`

Functions

- `template<typename _CharT, typename _Traits, typename _Alloc1 >`
`void std::tr2::dynamic_bitset< _WordT, _Alloc >::M_copy_to_string (std::basic_string< _CharT, _Traits, ↵`
`_Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const`
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`
`std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`
`const dynamic_bitset< _WordT, _Alloc > &__x)`
- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`
`std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__is,`
`dynamic_bitset< _WordT, _Alloc > &__x)`
- `template<typename _WordT, typename _Alloc >`
`bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, ↵`
`_Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc`
`> &__rhs)`
- `template<typename _WordT, typename _Alloc >`
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, ↵`
`_Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const`
`dynamic_bitset< _WordT, _Alloc > &__y)`

- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator|` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)
- `template<typename _WordT, typename _Alloc >`
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator-` (const `dynamic_bitset< _WordT, _Alloc > &__x`, const `dynamic_bitset< _WordT, _Alloc > &__y`)

2.30.1 Detailed Description

2.30.2 Function Documentation

2.30.2.1 `operator!=()`

```
template<typename _WordT, typename _Alloc >
bool std::tr2::operator!= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1123 of file `dynamic_bitset`.

2.30.2.2 `operator&()`

```
template<typename _WordT, typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator& (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1158 of file `dynamic_bitset`.

2.30.2.3 `operator-()`

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator- (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

Parameters

<code>__x</code>	A bitset.
<code>__y</code>	A bitset of the same size as <code>__x</code> .

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1188 of file `dynamic_bitset`.

2.30.2.4 `operator<<()`

```
template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
std::basic_ostream<_CharT, _Traits>& std::tr2::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const dynamic_bitset< _WordT, _Alloc > & __x ) [inline]
```

Stream output operator for `dynamic_bitset`.

Definition at line 1201 of file `dynamic_bitset`.

2.30.2.5 `operator<=()`

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator<= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1129 of file `dynamic_bitset`.

2.30.2.6 operator>()

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator> (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1135 of file dynamic_bitset.

2.30.2.7 operator>=()

```
template<typename _WordT , typename _Alloc >
bool std::tr2::operator>= (
    const dynamic_bitset< _WordT, _Alloc > & __lhs,
    const dynamic_bitset< _WordT, _Alloc > & __rhs ) [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1141 of file dynamic_bitset.

2.30.2.8 operator>>()

```
template<typename _CharT , typename _Traits , typename _WordT , typename _Alloc >
std::basic_istream<_CharT, _Traits>& std::tr2::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    dynamic_bitset< _WordT, _Alloc > & __x )
```

Stream input operator for dynamic_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

Definition at line 207 of file dynamic_bitset.tcc.

References std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::tr2::dynamic_bitset< _WordT, _Alloc >::size(), and std::basic_ios< _CharT, _Traits >::widen().

2.30.2.9 operator^()

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator^ (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

Parameters

$_x$	A bitset.
$_y$	A bitset of the same size as $_x$.

Returns

A new bitset.

These should be self-explanatory.

Definition at line 1178 of file dynamic_bitset.

2.30.2.10 operator" | ()

```
template<typename _WordT , typename _Alloc >
dynamic_bitset<_WordT, _Alloc> std::tr2::operator| (
    const dynamic_bitset< _WordT, _Alloc > & __x,
    const dynamic_bitset< _WordT, _Alloc > & __y ) [inline]
```

Global bitwise operations on bitsets.

Parameters

$_x$	A bitset.
$_y$	A bitset of the same size as $_x$.

Returns

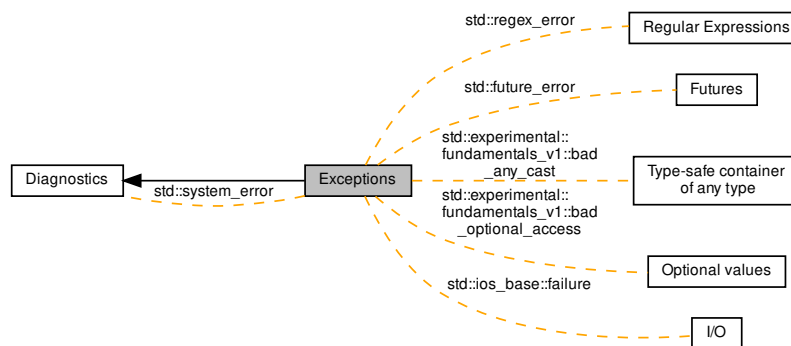
A new bitset.

These should be self-explanatory.

Definition at line 1168 of file dynamic_bitset.

2.31 Exceptions

Collaboration diagram for Exceptions:



Classes

- class `__cxxabiv1::__forced_unwind`
- class `std::bad_alloc`
- class `std::experimental::fundamentals_v1::bad_any_cast`
- class `std::bad_cast`
- class `std::bad_exception`
- class `std::bad_function_call`
- class `std::experimental::fundamentals_v1::bad_optional_access`
- class `std::bad_typeid`
- class `std::bad_weak_ptr`
- class `std::domain_error`
- class `std::exception`
- class `std::__exception_ptr::exception_ptr`
- class `std::ios_base::failure`
- struct `__gnu_cxx::forced_error`
- class `std::future_error`
- class `std::invalid_argument`
- class `std::length_error`
- class `std::logic_error`
- class `std::nested_exception`
- class `std::out_of_range`
- class `std::overflow_error`
- class `std::range_error`
- class `__gnu_cxx::recursive_init_error`
- class `std::regex_error`
- class `std::runtime_error`
- class `std::system_error`
- class `std::underflow_error`

Macros

- `#define __cpp_lib_uncaught_exceptions`

Typedefs

- `typedef void(* std::terminate_handler) ()`
- `typedef void(* std::unexpected_handler) ()`

Functions

- `std::exception::exception (const exception &)=default`
- `std::exception::exception (exception &&)=default`
- `void __gnu_cxx::__verbose_terminate_handler ()`
- `exception_ptr std::current_exception () noexcept`
- `terminate_handler std::get_terminate () noexcept`
- `unexpected_handler std::get_unexpected () noexcept`
- `template<typename _Ex >`
`exception_ptr std::make_exception_ptr (_Ex __ex) noexcept`
- `exception & std::exception::operator= (const exception &)=default`
- `exception & std::exception::operator= (exception &&)=default`
- `void std::rethrow_exception (exception_ptr)`
- `template<typename _Ex >`
`void std::rethrow_if_nested (const _Ex &__ex)`
- `terminate_handler std::set_terminate (terminate_handler) noexcept`
- `unexpected_handler std::set_unexpected (unexpected_handler) noexcept`
- `void std::terminate () noexcept`
- `template<typename _Tp >`
`void std::throw_with_nested (_Tp &&__t)`
- `bool std::uncaught_exception () noexcept`
- `int std::uncaught_exceptions () noexcept`
- `void std::unexpected ()`
- `virtual const char * std::exception::what () const noexcept`

2.31.1 Detailed Description

Classes and functions for reporting errors via exception classes.

2.31.2 Typedef Documentation

2.31.2.1 terminate_handler

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

Definition at line 65 of file exception.

2.31.2.2 unexpected_handler

```
typedef void(* std::unexpected_handler) ()
```

If you write a replacement unexpected handler, it must be of this type.

Definition at line 68 of file exception.

2.31.3 Function Documentation

2.31.3.1 __verbose_terminate_handler()

```
void __gnu_cxx::__verbose_terminate_handler ( )
```

A replacement for the standard terminate_handler which prints more information about the terminating exception (if any) on stderr.

Call

```
std::set_terminate(__gnu_cxx::__verbose_terminate_handler)
```

to use. For more info, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.html>.

In 3.4 and later, this is on by default.

2.31.3.2 current_exception()

```
exception_ptr std::current_exception ( ) [noexcept]
```

Obtain an exception_ptr to the currently handled exception. If there is none, or the currently handled exception is foreign, return the null value.

2.31.3.3 get_terminate()

```
terminate_handler std::get_terminate ( ) [noexcept]
```

Return the current terminate handler.

2.31.3.4 get_unexpected()

```
unexpected_handler std::get_unexpected ( ) [noexcept]
```

Return the current unexpected handler.

2.31.3.5 make_exception_ptr()

```
template<typename _Ex >  
exception_ptr std::make_exception_ptr (   
    _Ex __ex ) [noexcept]
```

Obtain an exception_ptr pointing to a copy of the supplied object.

Definition at line 186 of file exception_ptr.h.

2.31.3.6 rethrow_exception()

```
void std::rethrow_exception (   
    exception_ptr )
```

Throw the object pointed to by the exception_ptr.

2.31.3.7 rethrow_if_nested()

```
template<typename _Ex >  
void std::rethrow_if_nested (   
    const _Ex & __ex ) [inline]
```

If __ex is derived from nested_exception, __ex.rethrow_nested().

Definition at line 159 of file nested_exception.h.

References std::__addressof().

2.31.3.8 set_terminate()

```
terminate_handler std::set_terminate (   
    terminate_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

2.31.3.9 set_unexpected()

```
unexpected_handler std::set_unexpected (
    unexpected_handler ) [noexcept]
```

Takes a new handler function as an argument, returns the old function.

2.31.3.10 terminate()

```
void std::terminate ( ) [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

2.31.3.11 throw_with_nested()

```
template<typename _Tp >
void std::throw_with_nested (
    _Tp && __t ) [inline]
```

If `__t` is derived from `nested_exception`, throws `__t`. Else, throws an implementation-defined object derived from both.

Definition at line 118 of file `nested_exception.h`.

2.31.3.12 uncaught_exception()

```
bool std::uncaught_exception ( ) [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of `terminate()` (15.5.1).'

2.31.3.13 uncaught_exceptions()

```
int std::uncaught_exceptions ( ) [noexcept]
```

The number of uncaught exceptions.

2.31.3.14 unexpected()

```
void std::unexpected ( )
```

The runtime will call this function if an exception is thrown which violates the function's exception specification.

2.31.3.15 what()

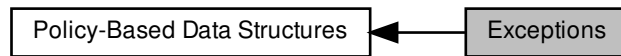
```
virtual const char* std::exception::what ( ) const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented in `std::experimental::filesystem::v1::filesystem_error`, `std::ios_base::failure`, `std::runtime_error`, `std::bad_typeid`, `std::bad_cast`, `std::logic_error`, `std::future_error`, `std::bad_weak_ptr`, `std::bad_alloc`, `std::experimental::fundamentals_v1::bad_exception`, and `std::bad_function_call`.

2.32 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct [__gnu_pbds::container_error](#)
- struct [__gnu_pbds::insert_error](#)
- struct [__gnu_pbds::join_error](#)
- struct [__gnu_pbds::resize_error](#)

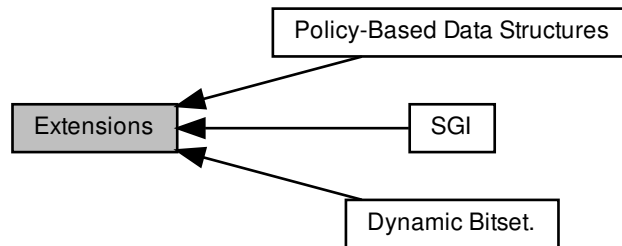
Functions

- void [__gnu_pbds::__throw_container_error\(\)](#)
- void [__gnu_pbds::__throw_insert_error\(\)](#)
- void [__gnu_pbds::__throw_join_error\(\)](#)
- void [__gnu_pbds::__throw_resize_error\(\)](#)

2.32.1 Detailed Description

2.33 Extensions

Collaboration diagram for Extensions:



Modules

- [Dynamic Bitset.](#)
- [Policy-Based Data Structures](#)
- [SGI](#)

Namespaces

- [__gnu_cxx](#)
- [std::tr2](#)

Classes

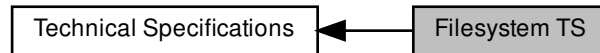
- [class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >](#)

2.33.1 Detailed Description

Components generally useful that are not part of any standard.

2.34 Filesystem TS

Collaboration diagram for Filesystem TS:



Files

- file [experimental/filesystem](#)

Classes

- class [std::experimental::filesystem::v1::filesystem_error](#)
- class [std::experimental::filesystem::v1::path::iterator](#)
- class [std::experimental::filesystem::v1::path](#)

Typedefs

- using **`std::experimental::filesystem::v1::file_time_type`** = `std::chrono::system_clock::time_point`

Enumerations

- enum [std::experimental::filesystem::v1::copy_options](#) : unsigned short { **`none`**, **`skip_existing`**, **`overwrite_existing`**, **`update_existing`**, **`recursive`**, **`copy_symlinks`**, **`skip_symlinks`**, **`directories_only`**, **`create_symlinks`**, **`create_hard_links`** }
- enum **`directory_options`** : unsigned char { **`none`**, **`follow_directory_symlink`**, **`skip_permission_denied`** }
- enum **`file_type`** : signed char { **`none`**, **`not_found`**, **`regular`**, **`directory`**, **`symlink`**, **`block`**, **`character`**, **`fifo`**, **`socket`**, **`unknown`** }
- enum [std::experimental::filesystem::v1::perms](#) : unsigned { **`none`**, **`owner_read`**, **`owner_write`**, **`owner_exec`**, **`owner_all`**, **`group_read`**, **`group_write`**, **`group_exec`**, **`group_all`**, **`others_read`**, **`others_write`**, **`others_exec`**, **`others_all`**, **`all`**, **`set_uid`**, **`set_gid`**, **`sticky_bit`**, **`mask`**, **`unknown`**, **`add_perms`**, **`remove_perms`**, **`symlink_nofollow`** }

Functions

- `path std::experimental::filesystem::v1::absolute` (const `path` &__p, const `path` &__base=current_path())
- `path & std::experimental::filesystem::v1::path::assign` (`string_type` &&__source)
- `directory_iterator std::experimental::filesystem::v1::begin` (`directory_iterator` __iter) noexcept
- `recursive_directory_iterator std::experimental::filesystem::v1::begin` (`recursive_directory_iterator` __iter) noexcept
- `iterator std::experimental::filesystem::v1::path::begin` () const
- `path std::experimental::filesystem::v1::canonical` (const `path` &__p, const `path` &__base=current_path())
- `path std::experimental::filesystem::v1::canonical` (const `path` &__p, `error_code` &__ec)
- `path std::experimental::filesystem::v1::canonical` (const `path` &__p, const `path` &__base, `error_code` &__ec)
- `int std::experimental::filesystem::v1::path::compare` (const `string_type` &__s) const
- `int std::experimental::filesystem::v1::path::compare` (const `value_type` *__s) const
- `int std::experimental::filesystem::v1::path::compare` (const `basic_string_view`< `value_type` > __s) const
- `void std::experimental::filesystem::v1::copy` (const `path` &__from, const `path` &__to)
- `void std::experimental::filesystem::v1::copy` (const `path` &__from, const `path` &__to, `error_code` &__ec) noexcept
- `void std::experimental::filesystem::v1::copy` (const `path` &__from, const `path` &__to, `copy_options` __options)
- `void std::experimental::filesystem::v1::copy` (const `path` &__from, const `path` &__to, `copy_options` __options, `error_code` &) noexcept
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &__from, const `path` &__to)
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &__from, const `path` &__to, `error_code` &__ec) noexcept
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &__from, const `path` &__to, `copy_options` __option) option)
- `bool std::experimental::filesystem::v1::copy_file` (const `path` &__from, const `path` &__to, `copy_options` __option, `error_code` &) noexcept
- `void std::experimental::filesystem::v1::copy_symlink` (const `path` &__existing_symlink, const `path` &__new<←__symlink)
- `void std::experimental::filesystem::v1::copy_symlink` (const `path` &__existing_symlink, const `path` &__new<←__symlink, `error_code` &__ec) noexcept
- `bool std::experimental::filesystem::v1::create_directories` (const `path` &__p)
- `bool std::experimental::filesystem::v1::create_directories` (const `path` &__p, `error_code` &__ec) noexcept
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &__p)
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &__p, `error_code` &__ec) noexcept
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &__p, const `path` &attributes)
- `bool std::experimental::filesystem::v1::create_directory` (const `path` &__p, const `path` &attributes, `error_code` &__ec) noexcept
- `void std::experimental::filesystem::v1::create_directory_symlink` (const `path` &__to, const `path` &__new<←symlink)
- `void std::experimental::filesystem::v1::create_directory_symlink` (const `path` &__to, const `path` &__new<←symlink, `error_code` &__ec) noexcept
- `void std::experimental::filesystem::v1::create_hard_link` (const `path` &__to, const `path` &__new_hard_link)
- `void std::experimental::filesystem::v1::create_hard_link` (const `path` &__to, const `path` &__new_hard_link, `error_code` &__ec) noexcept
- `void std::experimental::filesystem::v1::create_symlink` (const `path` &__to, const `path` &__new_symlink)
- `void std::experimental::filesystem::v1::create_symlink` (const `path` &__to, const `path` &__new_symlink, `error_code` &__ec) noexcept
- `path std::experimental::filesystem::v1::current_path` (`error_code` &__ec)
- `void std::experimental::filesystem::v1::current_path` (const `path` &__p)
- `void std::experimental::filesystem::v1::current_path` (const `path` &__p, `error_code` &__ec) noexcept
- `path std::experimental::filesystem::v1::current_path` ()

- directory_iterator **std::experimental::filesystem::v1::end** (directory_iterator) noexcept
- recursive_directory_iterator **std::experimental::filesystem::v1::end** (recursive_directory_iterator) noexcept
- iterator **std::experimental::filesystem::v1::path::end** () const
- bool **std::experimental::filesystem::v1::equivalent** (const path &__p1, const path &__p2)
- bool **std::experimental::filesystem::v1::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::exists** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::exists** (const path &__p)
- bool **std::experimental::filesystem::v1::exists** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::path::extension** () const
- uintmax_t **std::experimental::filesystem::v1::file_size** (const path &__p)
- uintmax_t **std::experimental::filesystem::v1::file_size** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::path::filename** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
std::basic_string<_CharT, _Traits, _Allocator> **std::experimental::filesystem::v1::path::generic_string**
(const _Allocator &__a= _Allocator()) const
- std::string **std::experimental::filesystem::v1::path::generic_string** () const
- std::u16string **std::experimental::filesystem::v1::path::generic_u16string** () const
- std::u32string **std::experimental::filesystem::v1::path::generic_u32string** () const
- std::string **std::experimental::filesystem::v1::path::generic_u8string** () const
- std::wstring **std::experimental::filesystem::v1::path::generic_wstring** () const
- uintmax_t **std::experimental::filesystem::v1::hard_link_count** (const path &__p)
- uintmax_t **std::experimental::filesystem::v1::hard_link_count** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::path::has_extension** () const
- bool **std::experimental::filesystem::v1::path::has_stem** () const
- bool **std::experimental::filesystem::v1::path::is_absolute** () const
- bool **std::experimental::filesystem::v1::is_block_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_block_file** (const path &__p)
- bool **std::experimental::filesystem::v1::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_character_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_character_file** (const path &__p)
- bool **std::experimental::filesystem::v1::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_directory** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_directory** (const path &__p)
- bool **std::experimental::filesystem::v1::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_empty** (const path &__p)
- bool **std::experimental::filesystem::v1::is_empty** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_fifo** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_fifo** (const path &__p)
- bool **std::experimental::filesystem::v1::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_other** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_other** (const path &__p)
- bool **std::experimental::filesystem::v1::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_regular_file** (const path &__p)
- bool **std::experimental::filesystem::v1::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_regular_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_socket** (file_status __s) noexcept
- bool **std::experimental::filesystem::v1::is_socket** (const path &__p)
- bool **std::experimental::filesystem::v1::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::is_symlink** (const path &__p)
- bool **std::experimental::filesystem::v1::is_symlink** (const path &__p, error_code &__ec) noexcept

- `bool std::experimental::filesystem::v1::is_symlink (file_status) noexcept`
- `file_time_type std::experimental::filesystem::v1::last_write_time (const path &__p)`
- `file_time_type std::experimental::filesystem::v1::last_write_time (const path &__p, error_code &__ec) noexcept`
- `void std::experimental::filesystem::v1::last_write_time (const path &__p, file_time_type __new_time)`
- `void std::experimental::filesystem::v1::last_write_time (const path &__p, file_time_type __new_time, error_code &__ec) noexcept`
- `path & std::experimental::filesystem::v1::path::make_preferred ()`
- `bool std::experimental::filesystem::v1::operator!= (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::v1::operator!= (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `constexpr copy_options std::experimental::filesystem::v1::operator& (copy_options __x, copy_options __y) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator& (perms __x, perms __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::v1::operator& (directory_options __x, directory_options __y) noexcept`
- `copy_options & std::experimental::filesystem::v1::operator&= (copy_options &__x, copy_options __y) noexcept`
- `perms & std::experimental::filesystem::v1::operator&= (perms &__x, perms __y) noexcept`
- `directory_options & std::experimental::filesystem::v1::operator&= (directory_options &__x, directory_options __y) noexcept`
- `reference std::experimental::filesystem::v1::path::iterator::operator* () const`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator++ ()`
- `path & std::experimental::filesystem::v1::path::operator+= (const path &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const string_type &__x)`
- `path & std::experimental::filesystem::v1::path::operator+= (const value_type * __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (value_type __x)`
- `path & std::experimental::filesystem::v1::path::operator+= (basic_string_view< value_type > __x)`
- `template<typename _CharT >
__detail::Path< _CharT *, _CharT * > & std::experimental::filesystem::v1::path::operator+= (_CharT __x)`
- `iterator & std::experimental::filesystem::v1::path::iterator::operator-- ()`
- `path & std::experimental::filesystem::v1::path::operator= (path &&__p) noexcept`
- `path & std::experimental::filesystem::v1::path::operator= (string_type &&__source)`
- `bool std::experimental::filesystem::v1::operator== (const directory_iterator &__lhs, const directory_iterator &__rhs)`
- `bool std::experimental::filesystem::v1::operator== (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)`
- `constexpr copy_options std::experimental::filesystem::v1::operator^ (copy_options __x, copy_options __y) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator^ (perms __x, perms __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::v1::operator^ (directory_options __x, directory_options __y) noexcept`
- `copy_options & std::experimental::filesystem::v1::operator^= (copy_options &__x, copy_options __y) noexcept`
- `perms & std::experimental::filesystem::v1::operator^= (perms &__x, perms __y) noexcept`
- `directory_options & std::experimental::filesystem::v1::operator^= (directory_options &__x, directory_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::v1::operator| (copy_options __x, copy_options __y) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator| (perms __x, perms __y) noexcept`

- constexpr directory_options **std::experimental::filesystem::v1::operator|** (directory_options __x, directory_options __y) noexcept
- copy_options & **std::experimental::filesystem::v1::operator|**= (copy_options &__x, copy_options __y) noexcept
- perms & **std::experimental::filesystem::v1::operator|**= (perms &__x, perms __y) noexcept
- directory_options & **std::experimental::filesystem::v1::operator|**= (directory_options &__x, directory_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::v1::operator~** (copy_options __x) noexcept
- constexpr perms **std::experimental::filesystem::v1::operator~** (perms __x) noexcept
- constexpr directory_options **std::experimental::filesystem::v1::operator~** (directory_options __x) noexcept
- void **std::experimental::filesystem::v1::permissions** (const path &__p, perms __prms)
- void **std::experimental::filesystem::v1::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::read_symlink** (const path &__p)
- path **std::experimental::filesystem::v1::read_symlink** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::v1::remove** (const path &__p)
- bool **std::experimental::filesystem::v1::remove** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::v1::remove_all** (const path &__p)
- uintmax_t **std::experimental::filesystem::v1::remove_all** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::v1::rename** (const path &__from, const path &__to)
- void **std::experimental::filesystem::v1::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::experimental::filesystem::v1::resize_file** (const path &__p, uintmax_t __size)
- void **std::experimental::filesystem::v1::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::experimental::filesystem::v1::space** (const path &__p)
- space_info **std::experimental::filesystem::v1::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::experimental::filesystem::v1::status** (const path &__p)
- file_status **std::experimental::filesystem::v1::status** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::status_known** (file_status) noexcept
- path **std::experimental::filesystem::v1::path::stem** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> **std::basic_string**<_CharT, _Traits, _Allocator> **std::experimental::filesystem::v1::path::string** (const _CharT* __a, _Allocator& __a=_Allocator()) const
- **std::string** **std::experimental::filesystem::v1::path::string** () const
- void **std::experimental::filesystem::v1::path::swap** (path &__rhs) noexcept
- file_status **std::experimental::filesystem::v1::symlink_status** (const path &__p)
- file_status **std::experimental::filesystem::v1::symlink_status** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::system_complete** (const path &__p)
- path **std::experimental::filesystem::v1::system_complete** (const path &__p, error_code &__ec)
- path **std::experimental::filesystem::v1::temp_directory_path** ()
- path **std::experimental::filesystem::v1::temp_directory_path** (error_code &__ec)
- **std::u16string** **std::experimental::filesystem::v1::path::u16string** () const
- **std::u32string** **std::experimental::filesystem::v1::path::u32string** () const
- **std::string** **std::experimental::filesystem::v1::path::u8string** () const
- **std::wstring** **std::experimental::filesystem::v1::path::wstring** () const

2.34.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories.

ISO/IEC TS 18822:2015 C++ File System Technical Specification

2.34.2 Enumeration Type Documentation

2.34.2.1 copy_options

```
enum std::experimental::filesystem::v1::copy_options : unsigned short [strong]
```

Bitmask type.

Definition at line 88 of file experimental/bits/fs_fwd.h.

2.34.2.2 perms

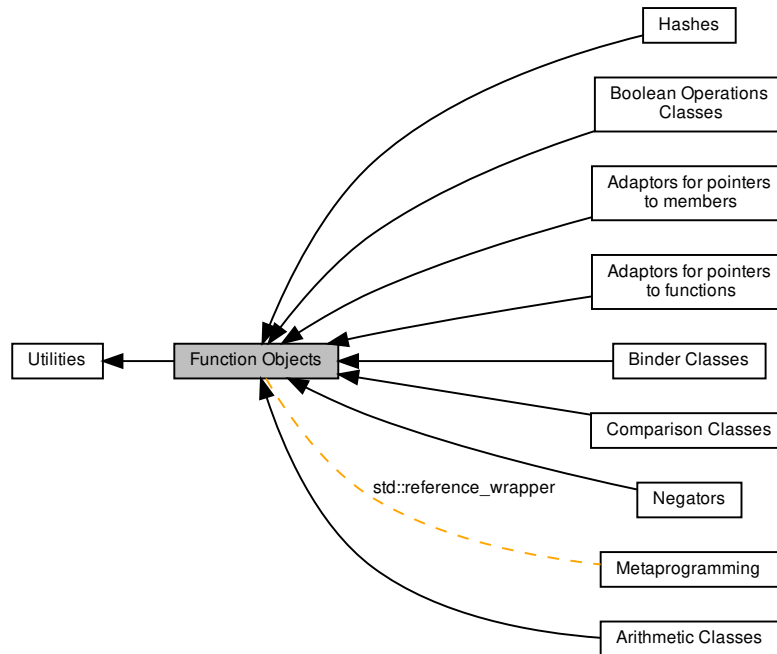
```
enum std::experimental::filesystem::v1::perms : unsigned [strong]
```

Bitmask type.

Definition at line 141 of file experimental/bits/fs_fwd.h.

2.35 Function Objects

Collaboration diagram for Function Objects:



Modules

- [Adaptors for pointers to functions](#)
- [Adaptors for pointers to members](#)
- [Arithmetic Classes](#)
- [Binder Classes](#)
- [Boolean Operations Classes](#)
- [Comparison Classes](#)
- [Hashes](#)
- [Negators](#)

Classes

- `struct std::binary_function< _Arg1, _Arg2, _Result >`
- `class std::function< _Res(_ArgTypes...)>`
- `class std::reference_wrapper< _Tp >`
- `struct std::unary_function< _Arg, _Result >`

Functions

- `template<typename _Tp, typename _Class >`
`constexpr _Mem_fn<_Tp _Class::*> std::mem_fn (_Tp _Class::* __pm) noexcept`

2.35.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.

Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables.

Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform(a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will be inlined directly.

The standard functors are derived from structs named `unary_function` and `binary_function`. These two classes contain nothing but typedefs, to aid in generic (template) programming. If you write your own functors, you might consider doing the same.

2.35.2 Function Documentation

2.35.2.1 `mem_fn()`

```
template<typename _Tp, typename _Class >
constexpr _Mem_fn<_Tp _Class::*> std::mem_fn (
    _Tp _Class::* __pm ) [inline], [noexcept]
```

Returns a function object that forwards to the member pointer `pm`.

Definition at line 170 of file `functional`.

2.36 Futures

Collaboration diagram for Futures:



Classes

- class `std::__basic_future< _Res >`
- struct `std::__future_base`
- struct `std::__future_base::_Result< _Res >`
- struct `std::__future_base::_Result< _Res & >`
- struct `std::__future_base::_Result< void >`
- struct `std::__future_base::_Result_alloc< _Res, _Alloc >`
- struct `std::__future_base::_Result_base`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

Typedefs

- template<typename `_Fn`, typename... `_Args`>
using `std::__async_result_of` = typename `__invoke_result< typename decay< _Fn >::type, typename decay< _Args >::type... >::type`

Enumerations

- enum `std::future_errc` { `future_already_retrieved`, `promise_already_satisfied`, `no_state`, `broken_promise` }
- enum `std::future_status` { `ready`, `timeout`, `deferred` }
- enum `std::launch` { `async`, `deferred` }

Functions

- `std::__basic_future< _Res >::__basic_future` (const `shared_future< _Res >` &) noexcept
- `std::__basic_future< _Res >::__basic_future` (`shared_future< _Res >` &&) noexcept
- `std::__basic_future< _Res >::__basic_future` (`future< _Res >` &&) noexcept
- `template<typename _Signature, typename _Fn, typename _Alloc = std::allocator<int>>>`
`static shared_ptr< __future_base::__Task_state_base< _Signature > > std::__create_task_state` (`_Fn` && `__fn`,
`const _Alloc` & `__a` = `_Alloc`())
- `template<typename _BoundFn >`
`static std::shared_ptr< _State_base > std::__future_base::__S_make_async_state` (`_BoundFn` && `__fn`)
- `template<typename _BoundFn >`
`static std::shared_ptr< _State_base > std::__future_base::__S_make_deferred_state` (`_BoundFn` && `__fn`)
- `template<typename _Fn, typename... _Args>`
`future< __async_result_of< _Fn, _Args... > > std::async` (`launch` `__policy`, `_Fn` && `__fn`, `_Args` &&... `__args`)
- `template<typename _Fn, typename... _Args>`
`future< __async_result_of< _Fn, _Args... > > std::async` (`_Fn` && `__fn`, `_Args` &&... `__args`)
- `const error_category & std::future_category` () noexcept
- `error_code std::make_error_code` (`future_errc` `__errc`) noexcept
- `error_condition std::make_error_condition` (`future_errc` `__errc`) noexcept
- `constexpr launch std::operator&` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator&=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator^` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator^=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator|` (`launch` `__x`, `launch` `__y`)
- `launch & std::operator|=` (`launch` & `__x`, `launch` `__y`)
- `constexpr launch std::operator~` (`launch` `__x`)
- `shared_future< _Res > std::future< _Res >::share` () noexcept
- `shared_future< _Res & > std::future< _Res & >::share` () noexcept
- `shared_future< void > std::future< void >::share` () noexcept
- `template<typename _Res >`
`void std::swap` (`promise< _Res >` & `__x`, `promise< _Res >` & `__y`) noexcept
- `template<typename _Res, typename... _ArgTypes>`
`void std::swap` (`packaged_task< _Res(_ArgTypes...) >` & `__x`, `packaged_task< _Res(_ArgTypes...) >` & `__y`) noexcept

2.36.1 Detailed Description

Classes for futures support.

2.36.2 Enumeration Type Documentation

2.36.2.1 future_errc

```
enum std::future_errc [strong]
```

Error code for futures.

Definition at line 66 of file future.

2.36.2.2 future_status

```
enum std::future_status [strong]
```

Status code for futures.

Definition at line 174 of file future.

2.36.2.3 launch

```
enum std::launch [strong]
```

Launch code for futures.

Definition at line 137 of file future.

2.36.3 Function Documentation

2.36.3.1 async() [1/2]

```
template<typename _Fn , typename... _Args>
future< __async_result_of< _Fn, _Args...  > > std::async (
    launch __policy,
    _Fn && __fn,
    _Args &&... __args )
```

async

Definition at line 1727 of file future.

2.36.3.2 async() [2/2]

```
template<typename _Fn , typename... _Args>
future< __async_result_of< _Fn, _Args...  > > std::async (
    _Fn && __fn,
    _Args &&... __args ) [inline]
```

async, potential overload

Definition at line 1760 of file future.

2.36.3.3 future_category()

```
const error_category& std::future_category ( ) [noexcept]
```

Points to a statically-allocated object derived from `error_category`.

2.36.3.4 make_error_code()

```
error_code std::make_error_code (
    future_errc __errc ) [inline], [noexcept]
```

Overload for `make_error_code`.

Definition at line 84 of file `future`.

2.36.3.5 make_error_condition()

```
error_condition std::make_error_condition (
    future_errc __errc ) [inline], [noexcept]
```

Overload for `make_error_condition`.

Definition at line 89 of file `future`.

2.36.3.6 swap()

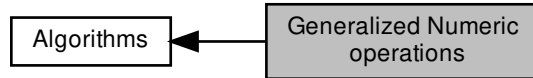
```
template<typename _Res , typename... _ArgTypes>
void std::swap (
    packaged_task< _Res(_ArgTypes...)> & __x,
    packaged_task< _Res(_ArgTypes...)> & __y ) [inline], [noexcept]
```

`swap`

Definition at line 1591 of file `future`.

2.37 Generalized Numeric operations

Collaboration diagram for Generalized Numeric operations:



Functions

- `template<typename _InputIterator, typename _Tp >`
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↔
binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`
`constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↔
__result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↔
__result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
__init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2`
`>`
`constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
__init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator >`
`constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`
`constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result,
_BinaryOperation __binary_op)`

2.37.1 Detailed Description

2.37.2 Function Documentation

2.37.2.1 accumulate() [1/2]

```
template<typename _InputIterator , typename _Tp >
constexpr _Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init ) [inline]
```

Accumulate values in a range.

Accumulates the values in the range `[first,last)` using `operator+()`. The initial value is *init*. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.

Returns

The final sum.

Definition at line 134 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_partial_sum_linear()`.

2.37.2.2 accumulate() [2/2]

```
template<typename _InputIterator , typename _Tp , typename _BinaryOperation >
constexpr _Tp std::accumulate (
    _InputIterator __first,
    _InputIterator __last,
    _Tp __init,
    _BinaryOperation __binary_op ) [inline]
```

Accumulate values in a range with operation.

Accumulates the values in the range `[first,last)` using the function object `__binary_op`. The initial value is `__init`. The values are processed in order.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op</code>	Function object to accumulate with.

Returns

The final sum.

Definition at line 161 of file `stl_numeric.h`.

2.37.2.3 adjacent_difference() [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
constexpr _OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[first,last)` using `operator-()` and writes the result to `__result`.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sums.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 338 of file `stl_numeric.h`.

2.37.2.4 adjacent_difference() [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
constexpr _OutputIterator std::adjacent_difference (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Return differences between adjacent values.

Computes the difference between adjacent values in the range `[__first,__last)` using the function object `__binary_op` and writes the result to `__result`.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

Returns

Iterator pointing just beyond the values written to result.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 539. `partial_sum` and `adjacent_difference` should mention requirements

Definition at line 382 of file `stl_numeric.h`.

2.37.2.5 `inner_product()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp >
constexpr _Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init ) [inline]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, multiplies successive elements from the two ranges and adds each product into the accumulated value using `operator+()`. The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.

Returns

The final inner product.

Definition at line 190 of file `stl_numeric.h`.

2.37.2.6 inner_product() [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Tp , typename _Binary↵
Operation1 , typename _BinaryOperation2 >
constexpr _Tp std::inner_product (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _Tp __init,
    _BinaryOperation1 __binary_op1,
    _BinaryOperation2 __binary_op2 ) [inline]
```

Compute inner product of two ranges.

Starting with an initial value of `__init`, applies `__binary_op2` to successive elements from the two ranges and accumulates each result into the accumulated value using `__binary_op1`. The values in the ranges are processed in order.

Parameters

<code>__first1</code>	Start of range 1.
<code>__last1</code>	End of range 1.
<code>__first2</code>	Start of range 2.
<code>__init</code>	Starting value to add other values to.
<code>__binary_op1</code>	Function object to accumulate with.
<code>__binary_op2</code>	Function object to apply to pairs of input values.

Returns

The final inner product.

Definition at line 223 of file `stl_numeric.h`.

2.37.2.7 iota()

```
template<typename _ForwardIterator , typename _Tp >
constexpr void std::iota (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Tp __value )
```

Create a range of sequentially increasing values.

For each element in the range `[first,last)` assigns `value` and increments `value` as if by `++value`.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__value</code>	Starting value.

Returns

Nothing.

Definition at line 88 of file `stl_numeric.h`.

2.37.2.8 `partial_sum()` [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
constexpr _OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result )
```

Return list of partial sums.

Accumulates the values in the range `[first,last)` using the `+` operator. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.

Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 256 of file `stl_numeric.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs_pu()`, and `__gnu_parallel::__sequential_random_shuffle()`.

2.37.2.9 `partial_sum()` [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryOperation >
constexpr _OutputIterator std::partial_sum (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Return list of partial sums.

Accumulates the values in the range `[first,last)` using `__binary_op`. As each successive input value is added into the total, that partial sum is written to `__result`. Therefore, the first value in `__result` is the first value of the input, the second value in `__result` is the sum of the first and second input values, and so on.

Parameters

<code>__first</code>	Start of input range.
<code>__last</code>	End of input range.
<code>__result</code>	Output sum.
<code>__binary_op</code>	Function object.

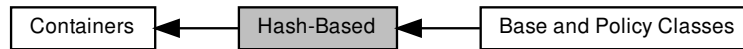
Returns

Iterator pointing just beyond the values written to `__result`.

Definition at line 298 of file `stl_numeric.h`.

2.38 Hash-Based

Collaboration diagram for Hash-Based:



Modules

- [Base and Policy Classes](#)

Classes

- [class `__gnu_pbds::basic_hash_table`](#)< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- [class `__gnu_pbds::cc_hash_table`](#)< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- [class `__gnu_pbds::gp_hash_table`](#)< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >

Macros

- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`

2.38.1 Detailed Description

2.39 Hashes

Collaboration diagram for Hashes:



Classes

- struct `std::hash< _Tp >`
- struct `std::hash< _Tp * >`
- struct `std::hash< bool >`
- struct `std::hash< char >`
- struct `std::hash< char16_t >`
- struct `std::hash< char32_t >`
- struct `std::hash< double >`
- struct `std::hash< float >`
- struct `std::hash< int >`
- struct `std::hash< long >`
- struct `std::hash< long double >`
- struct `std::hash< long long >`
- struct `std::hash< short >`
- struct `std::hash< signed char >`
- struct `std::hash< unsigned char >`
- struct `std::hash< unsigned int >`
- struct `std::hash< unsigned long >`
- struct `std::hash< unsigned long long >`
- struct `std::hash< unsigned short >`
- struct `std::hash< wchar_t >`

Macros

- `#define _Cxx_hashtable_define_trivial_hash(_Tp)`

2.39.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

2.40 Heap

Collaboration diagram for Heap:



Functions

- `template<typename _RandomAccessIterator >`
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.40.1 Detailed Description

2.40.2 Function Documentation

2.40.2.1 is_heap() [1/2]

```
template<typename _RandomAccessIterator >
constexpr bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Determines whether a range is a heap.

Parameters

<i>__first</i>	Start of range.
<i>__last</i>	End of range.

Returns

True if range is a heap, false otherwise.

Definition at line 550 of file `stl_heap.h`.

References `std::is_heap_until()`.

2.40.2.2 is_heap() [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr bool std::is_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Determines whether a range is a heap using comparison functor.

Parameters

<i>__first</i>	Start of range.
<i>__last</i>	End of range.
<i>__comp</i>	Comparison functor to use.

Returns

True if range is a heap, false otherwise.

Definition at line 564 of file `stl_heap.h`.

2.40.2.3 `is_heap_until()` [1/2]

```
template<typename _RandomAccessIterator >
constexpr _RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Search the end of a heap.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap.

Definition at line 496 of file `stl_heap.h`.

2.40.2.4 `is_heap_until()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr _RandomAccessIterator std::is_heap_until (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Search the end of a heap using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor to use.

Returns

An iterator pointing to the first element not in the heap.

This operation returns the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is a heap. Comparisons are made using `__comp`.

Definition at line 525 of file `stl_heap.h`.

Referenced by `std::is_heap()`.

2.40.2.5 make_heap() [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Construct a heap over a range.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation makes the elements in [`__first`,`__last`) into a heap.

Definition at line 374 of file `stl_heap.h`.

2.40.2.6 make_heap() [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::make_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Construct a heap over a range using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation makes the elements in [`__first`,`__last`) into a heap. Comparisons are made using `__comp`.

Definition at line 401 of file `stl_heap.h`.

2.40.2.7 pop_heap() [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Pop an element off a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

Precondition

`[__first, __last)` is a valid, non-empty range.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap.

Definition at line 282 of file `stl_heap.h`.

2.40.2.8 `pop_heap()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
constexpr void std::pop_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Pop an element off a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation pops the top of the heap. The elements `__first` and `__last-1` are swapped and `[__first, __last-1)` is made into a heap. Comparisons are made using `comp`.

Definition at line 316 of file `stl_heap.h`.

2.40.2.9 `push_heap()` [1/2]

```
template<typename _RandomAccessIterator>
constexpr void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Push an element onto a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.

This operation pushes the element at `last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap.

Definition at line 161 of file `stl_heap.h`.

2.40.2.10 `push_heap()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::push_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Push an element onto a heap using comparison functor.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap + element.
<code>__comp</code>	Comparison functor.

This operation pushes the element at `__last-1` onto the valid heap over the range `[__first,__last-1)`. After completion, `[__first,__last)` is a valid heap. Compare operations are performed using `comp`.

Definition at line 197 of file `stl_heap.h`.

Referenced by `std::priority_queue<_Tp, _Sequence, _Compare >::push()`.

2.40.2.11 `sort_heap()` [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort a heap.

Parameters

<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.

This operation sorts the valid heap in the range [`__first`,`__last`).

Definition at line 439 of file `stl_heap.h`.

2.40.2.12 `sort_heap()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::sort_heap (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort a heap using comparison functor.

Parameters

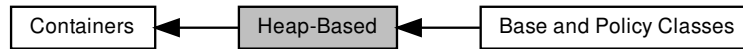
<code>__first</code>	Start of heap.
<code>__last</code>	End of heap.
<code>__comp</code>	Comparison functor to use.

This operation sorts the valid heap in the range [`__first`,`__last`). Comparisons are made using `__comp`.

Definition at line 467 of file `stl_heap.h`.

2.41 Heap-Based

Collaboration diagram for Heap-Based:



Modules

- [Base and Policy Classes](#)

Classes

- class [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>](#)

Typedefs

- typedef `_Alloc` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::allocator_type](#)
- typedef `Cmp_Fn` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::cmp_fn](#)
- typedef `base_type::const_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_iterator](#)
- typedef `__rebind_va::const_pointer` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_pointer](#)
- typedef `__rebind_va::const_reference` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::const_reference](#)
- typedef `Tag` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::container_category](#)
- typedef `allocator_type::difference_type` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::difference_type](#)
- typedef `base_type::iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::iterator](#)
- typedef `base_type::point_const_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::point_const_iterator](#)
- typedef `base_type::point_iterator` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::point_iterator](#)
- typedef `__rebind_va::pointer` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::pointer](#)
- typedef `__rebind_va::reference` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::reference](#)
- typedef `allocator_type::size_type` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::size_type](#)
- typedef `_Tv` [__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::value_type](#)

Functions

- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (const cmp_fn &r_cmp_fn)
- `template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (It first_it, It last_it)
- `template<typename It > __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue` (const `priority_queue` &other)
- `priority_queue` & `__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::operator=` (const `priority_queue` &other)
- `void __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::swap` (`priority_queue` &other)

2.41.1 Detailed Description

2.41.2 Function Documentation

2.41.2.1 `priority_queue()` [1/3]

```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue (
    const cmp_fn & r_cmp_fn ) [inline]
```

Constructor taking some policy objects. `r_cmp_fn` will be copied by the `Cmp_Fn` object of the container object.

Definition at line 115 of file `priority_queue.hpp`.

2.41.2.2 `priority_queue()` [2/3]

```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>::priority_queue (
    It first_it,
    It last_it ) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 121 of file `priority_queue.hpp`.

2.41.2.3 priority_queue() [3/3]

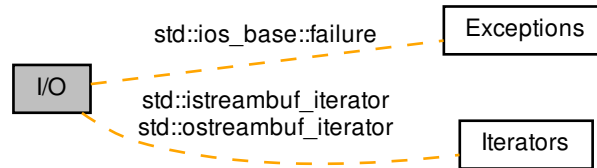
```
template<typename _Tv , typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
    It first_it,
    It last_it,
    const cmp_fn & r_cmp_fn ) [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.

Definition at line 129 of file priority_queue.hpp.

2.42 I/O

Collaboration diagram for I/O:



Classes

- class `std::basic_filebuf< _CharT, _Traits >`
- class `std::basic_fstream< _CharT, _Traits >`
- class `std::basic_ifstream< _CharT, _Traits >`
- class `std::basic_ios< _CharT, _Traits >`
- class `std::basic_iostream< _CharT, _Traits >`
- class `std::basic_istream< _CharT, _Traits >`
- class `std::basic_istreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_ofstream< _CharT, _Traits >`
- class `std::basic_ostream< _CharT, _Traits >`
- class `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_streambuf< _CharT, _Traits >`
- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`
- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`
- class `std::ios_base::failure`
- class `std::ios_base`
- class `std::istreambuf_iterator< _CharT, _Traits >`
- class `std::ostreambuf_iterator< _CharT, _Traits >`
- class `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`
- class `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`

Typedefs

- typedef `basic_filebuf< char >` `std::filebuf`
- typedef `basic_fstream< char >` `std::fstream`
- typedef `basic_ifstream< char >` `std::ifstream`
- typedef `basic_ios< char >` `std::ios`
- typedef `basic_iostream< char >` `std::iostream`
- typedef `basic_istream< char >` `std::istream`
- typedef `basic_istreamstream< char >` `std::istreamstream`
- typedef `basic_ofstream< char >` `std::ofstream`

- typedef `basic_ostream`< char > `std::ostream`
- typedef `basic_ostringstream`< char > `std::ostringstream`
- typedef `basic_streambuf`< char > `std::streambuf`
- typedef `basic_stringbuf`< char > `std::stringbuf`
- typedef `basic_stringstream`< char > `std::stringstream`
- typedef `basic_filebuf`< wchar_t > `std::wfilebuf`
- typedef `basic_fstream`< wchar_t > `std::wfstream`
- typedef `basic_ifstream`< wchar_t > `std::wifstream`
- typedef `basic_ios`< wchar_t > `std::wios`
- typedef `basic_iostream`< wchar_t > `std::wiostream`
- typedef `basic_istream`< wchar_t > `std::wistream`
- typedef `basic_istreamstream`< wchar_t > `std::wistreamstream`
- typedef `basic_ofstream`< wchar_t > `std::wofstream`
- typedef `basic_ostream`< wchar_t > `std::wostream`
- typedef `basic_ostringstream`< wchar_t > `std::wostringstream`
- typedef `basic_streambuf`< wchar_t > `std::wstreambuf`
- typedef `basic_stringbuf`< wchar_t > `std::wstringbuf`
- typedef `basic_stringstream`< wchar_t > `std::wstringstream`

2.42.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.

For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char> ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.

For more specific declarations, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.↵html#std.io.objects>

2.42.2 Typedef Documentation

2.42.2.1 filebuf

```
typedef basic_filebuf<char> std::filebuf
```

Class for `char` file buffers.

Definition at line 159 of file `iosfwd`.

2.42.2.2 fstream

```
typedef basic_fstream<char> std::fstream
```

Class for `char` mixed input and output file streams.

Definition at line 168 of file `iosfwd`.

2.42.2.3 ifstream

```
typedef basic_ifstream<char> std::ifstream
```

Class for `char` input file streams.

Definition at line 162 of file `iosfwd`.

2.42.2.4 ios

```
typedef basic_ios<char> std::ios
```

Base class for `char` streams.

Definition at line 128 of file `iosfwd`.

2.42.2.5 ostream

```
typedef basic_ostream<char> std::ostream
```

Base class for `char` mixed input and output streams.

Definition at line 144 of file `iosfwd`.

2.42.2.6 istream

```
typedef basic_istream<char> std::istream
```

Base class for `char` input streams.

Definition at line 138 of file `iosfwd`.

2.42.2.7 `istringstream`

```
typedef basic_istringstream<char> std::istringstream
```

Class for `char` input memory streams.

Definition at line 150 of file `iosfwd`.

2.42.2.8 `ofstream`

```
typedef basic_ofstream<char> std::ofstream
```

Class for `char` output file streams.

Definition at line 165 of file `iosfwd`.

2.42.2.9 `ostream`

```
typedef basic_ostream<char> std::ostream
```

Base class for `char` output streams.

Definition at line 141 of file `iosfwd`.

2.42.2.10 `ostringstream`

```
typedef basic_ostringstream<char> std::ostringstream
```

Class for `char` output memory streams.

Definition at line 153 of file `iosfwd`.

2.42.2.11 `streambuf`

```
typedef basic_streambuf<char> std::streambuf
```

Base class for `char` buffers.

Definition at line 135 of file `iosfwd`.

2.42.2.12 `stringbuf`

```
typedef basic_stringbuf<char> std::stringbuf
```

Class for `char` memory buffers.

Definition at line 147 of file `iosfwd`.

2.42.2.13 `stringstream`

```
typedef basic_stringstream<char> std::stringstream
```

Class for `char` mixed input and output memory streams.

Definition at line 156 of file `iosfwd`.

2.42.2.14 `wfilebuf`

```
typedef basic_filebuf<wchar_t> std::wfilebuf
```

Class for `wchar_t` file buffers.

Definition at line 199 of file `iosfwd`.

2.42.2.15 `wfstream`

```
typedef basic_fstream<wchar_t> std::wfstream
```

Class for `wchar_t` mixed input and output file streams.

Definition at line 208 of file `iosfwd`.

2.42.2.16 `wifstream`

```
typedef basic_ifstream<wchar_t> std::wifstream
```

Class for `wchar_t` input file streams.

Definition at line 202 of file `iosfwd`.

2.42.2.17 wios

```
typedef basic_ios<wchar_t> std::wios
```

Base class for `wchar_t` streams.

Definition at line 172 of file `iosfwd`.

2.42.2.18 wiostream

```
typedef basic_iostream<wchar_t> std::wiostream
```

Base class for `wchar_t` mixed input and output streams.

Definition at line 184 of file `iosfwd`.

2.42.2.19 wistream

```
typedef basic_istream<wchar_t> std::wistream
```

Base class for `wchar_t` input streams.

Definition at line 178 of file `iosfwd`.

2.42.2.20 wstringstream

```
typedef basic_istringstream<wchar_t> std::wstringstream
```

Class for `wchar_t` input memory streams.

Definition at line 190 of file `iosfwd`.

2.42.2.21 wofstream

```
typedef basic_ofstream<wchar_t> std::wofstream
```

Class for `wchar_t` output file streams.

Definition at line 205 of file `iosfwd`.

2.42.2.22 wostream

```
typedef basic_ostream<wchar_t> std::wostream
```

Base class for `wchar_t` output streams.

Definition at line 181 of file `iosfwd`.

2.42.2.23 wostringstream

```
typedef basic_ostringstream<wchar_t> std::wostringstream
```

Class for `wchar_t` output memory streams.

Definition at line 193 of file `iosfwd`.

2.42.2.24 wstreambuf

```
typedef basic_streambuf<wchar_t> std::wstreambuf
```

Base class for `wchar_t` buffers.

Definition at line 175 of file `iosfwd`.

2.42.2.25 wstringbuf

```
typedef basic_stringbuf<wchar_t> std::wstringbuf
```

Class for `wchar_t` memory buffers.

Definition at line 187 of file `iosfwd`.

2.42.2.26 wstringstream

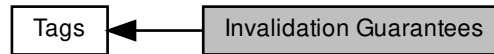
```
typedef basic_stringstream<wchar_t> std::wstringstream
```

Class for `wchar_t` mixed input and output memory streams.

Definition at line 196 of file `iosfwd`.

2.43 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



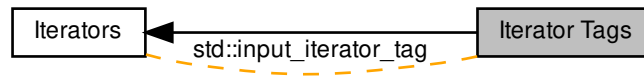
Classes

- struct [__gnu_pbds::basic_invalidation_guarantee](#)
- struct [__gnu_pbds::point_invalidation_guarantee](#)
- struct [__gnu_pbds::range_invalidation_guarantee](#)

2.43.1 Detailed Description

2.44 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

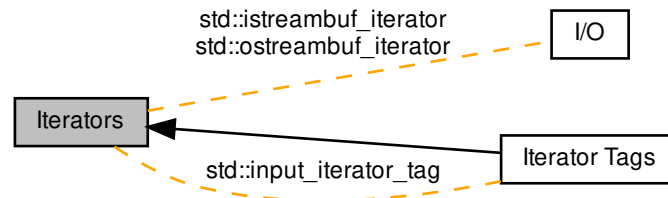
- struct [std::bidirectional_iterator_tag](#)
- struct [std::forward_iterator_tag](#)
- struct [std::input_iterator_tag](#)
- struct [std::output_iterator_tag](#)
- struct [std::random_access_iterator_tag](#)

2.44.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

2.45 Iterators

Collaboration diagram for Iterators:



Modules

- [Iterator Tags](#)

Classes

- class `std::back_insert_iterator< _Container >`
- class `std::front_insert_iterator< _Container >`
- struct `std::input_iterator_tag`
- class `std::insert_iterator< _Container >`
- class `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`
- class `std::istreambuf_iterator< _CharT, _Traits >`
- struct `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >`
- struct `std::iterator_traits< _Iterator >`
- struct `std::iterator_traits< _Tp * >`
- struct `std::iterator_traits< const _Tp * >`
- class `std::move_iterator< _Iterator >`
- class `std::ostream_iterator< _Tp, _CharT, _Traits >`
- class `std::ostreambuf_iterator< _CharT, _Traits >`
- class `std::reverse_iterator< _Iterator >`

Macros

- `#define __cpp_lib_make_reverse_iterator`

Functions

- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`
`copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__`
`copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT, typename _Size >`
`__enable_if_t< __is_char< _CharT >::__value, _CharT * > std::__`
`copy_n_a (istreambuf_iterator< _CharT > __it, _Size __n, _CharT * __result)`
- `template<typename _Iter >`
`constexpr iterator_traits< _Iter >::iterator_category std::__`
`iterator_category (const _Iter &)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond <typename iterator_traits<`
`_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`
`constexpr _ReturnType std::__`
`make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = typename conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move`
`_iterator<_Tp*>>::type>`
`constexpr _ReturnType std::__`
`make_move_if_noexcept_iterator (_Tp * __i)`
- `template<typename _Iterator >`
`constexpr reverse_iterator< _Iterator > std::__`
`make_reverse_iterator (_Iterator __i)`
- `template<typename _Iterator >`
`constexpr auto std::__`
`miter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(`
`__miter_base(__it.base())))`
- `template<typename _Iterator >`
`constexpr auto std::__`
`niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(`
`__niter_base(__it.base())))`
- `template<typename _CharT, typename _Distance >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type std::__`
`advance (istreambuf_iterator< _CharT > & __i, _Distance __n)`
- `template<typename _Container >`
`constexpr back_insert_iterator< _Container > std::__`
`back_inserter (_Container & __x)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__`
`copy (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`
`__result)`
- `template<typename _CharT >`
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::__`
`find (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _Container >`
`constexpr front_insert_iterator< _Container > std::__`
`front_inserter (_Container & __x)`
- `template<typename _Container >`
`insert_iterator< _Container > std::__`
`inserter (_Container & __x, typename _Container::iterator __i)`
- `template<typename _Iterator >`
`constexpr move_iterator< _Iterator > std::__`
`make_move_iterator (_Iterator __i)`
- `template<typename _Iterator >`
`constexpr reverse_iterator< _Iterator > std::__`
`make_reverse_iterator (_Iterator __i)`
- `template<typename _CharT, typename _Traits >`
`bool std::__`
`operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`
`_Traits > & __b)`

- `template<typename _Iterator >`
`constexpr bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator!= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`
`&__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator!= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`constexpr reverse_iterator< _Iterator > std::operator+ (typename reverse_iterator< _Iterator >::difference_↵`
`type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Iterator >`
`constexpr move_iterator< _Iterator > std::operator+ (typename move_iterator< _Iterator >::difference_type ↵`
`__n, const move_iterator< _Iterator > &__x)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr auto std::operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR >`
`&__y) -> decltype(__y.base() - __x.base())`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr auto std::operator- (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &↵`
`__y) -> decltype(__x.base() - __y.base())`
- `template<typename _Iterator >`
`constexpr bool std::operator< (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`
`&__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator< (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`
`&__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &↵`
`__y)`
- `template<typename _CharT, typename _Traits >`
`bool std::operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT,`
`_Traits > &__b)`
- `template<typename _Iterator >`
`constexpr bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`
`&__y)`

- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`
`constexpr bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`
`constexpr bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

2.45.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

2.45.2 Function Documentation

2.45.2.1 `__iterator_category()`

```
template<typename _Iter >
constexpr iterator\_traits<_Iter>::iterator_category std::__iterator_category (
    const _Iter & ) [inline]
```

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.

Definition at line 238 of file `stl_iterator_base_types.h`.

Referenced by `__gnu_debug::__valid_range_aux()`, `std::advance()`, `std::deque< _StateSeqT >::assign()`, `std::deque< _StateSeqT >::deque()`, `std::distance()`, `std::deque< _StateSeqT >::insert()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector()`.

2.45.2.2 back_inserter()

```
template<typename _Container >
constexpr back_insert_iterator<_Container> std::back_inserter (
    _Container & __x ) [inline]
```

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `back_insert_iterator` working on `__x`.

This wrapper function helps in creating `back_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 685 of file `bits/stl_iterator.h`.

Referenced by `std::match_results<_Bi_iter>::format()`, and `std::regex_replace()`.

2.45.2.3 front_inserter()

```
template<typename _Container >
constexpr front_insert_iterator<_Container> std::front_inserter (
    _Container & __x ) [inline]
```

Parameters

<code>__x</code>	A container of arbitrary type.
------------------	--------------------------------

Returns

An instance of `front_insert_iterator` working on `x`.

This wrapper function helps in creating `front_insert_iterator` instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 788 of file `bits/stl_iterator.h`.

2.45.2.4 inserter()

```
template<typename _Container >
insert_iterator<_Container> std::inserter (
    _Container & __x,
    typename _Container::iterator __i ) [inline]
```

Parameters

\leftarrow __x	A container of arbitrary type.
\leftarrow __i	An iterator into the container.

Returns

An instance of insert_iterator working on __x.

This wrapper function helps in creating insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

Definition at line 931 of file bits/stl_iterator.h.

2.45.2.5 make_reverse_iterator()

```
template<typename _Iterator >
constexpr reverse_iterator<_Iterator> std::make_reverse_iterator (
    _Iterator __i ) [inline]
```

Generator function for reverse_iterator.

Definition at line 552 of file bits/stl_iterator.h.

2.45.2.6 operator==()

```
template<typename _Iterator >
constexpr bool std::operator== (
    const reverse_iterator<_Iterator > & __x,
    const reverse_iterator<_Iterator > & __y ) [inline]
```

Parameters

\leftarrow __x	A reverse_iterator.
\leftarrow __y	A reverse_iterator.

Returns

A simple bool.

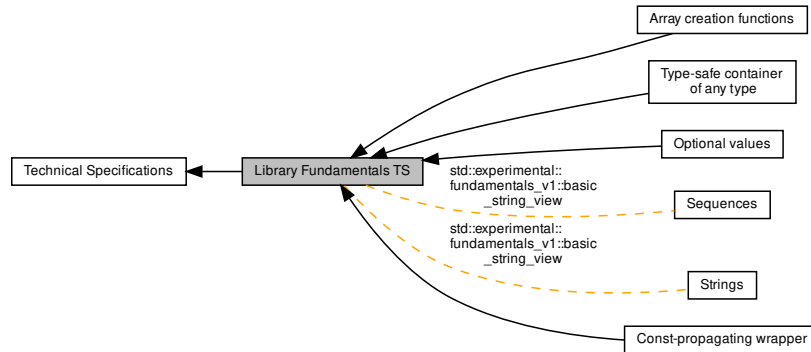
Reverse iterators forward comparisons to their underlying `base()` iterators.

Definition at line 385 of file `bits/stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::base()`.

2.46 Library Fundamentals TS

Collaboration diagram for Library Fundamentals TS:



Modules

- [Array creation functions](#)
- [Const-propagating wrapper](#)
- [Optional values](#)
- [Type-safe container of any type](#)

Files

- file [experimental/algorithm](#)
- file [experimental/any](#)
- file [experimental/array](#)
- file [experimental/chrono](#)
- file [experimental/deque](#)
- file [experimental/forward_list](#)
- file [experimental/functional](#)
- file [experimental/iterator](#)
- file [experimental/list](#)
- file [experimental/map](#)
- file [experimental/memory](#)
- file [experimental/memory_resource](#)
- file [experimental/numeric](#)
- file [experimental/optional](#)
- file [propagate_const](#)
- file [experimental/random](#)
- file [experimental/ratio](#)
- file [experimental/regex](#)
- file [experimental/set](#)
- file [experimental/string](#)

- file [experimental/string_view](#)
- file [experimental/system_error](#)
- file [experimental/tuple](#)
- file [experimental/type_traits](#)
- file [experimental/unordered_map](#)
- file [experimental/unordered_set](#)
- file [experimental/utility](#)
- file [experimental/vector](#)

Classes

- class [std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>](#)

2.46.1 Detailed Description

Components defined by the *C++ Extensions for Library Fundamentals* Technical Specification, versions 1 and 2.

- ISO/IEC TS 19568:2015 C++ Extensions for Library Fundamentals
- ISO/IEC TS 19568:2017 C++ Extensions for Library Fundamentals, Version 2

2.47 List-Based

Collaboration diagram for List-Based:



Classes

- class `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >`

Macros

- `#define PB_DS_LU_BASE`

2.47.1 Detailed Description

2.48 Locales

Classes

- class `std::codecvt< _InternT, _ExternT, _StateT >`
- class `std::ctype< _CharT >`
- class `std::ctype< char >`
- class `std::ctype< wchar_t >`
- class `std::locale::facet`
- class `std::locale::id`
- class `std::locale`
- class `std::messages< _CharT >`
- struct `std::messages_base`
- class `std::money_base`
- class `std::money_get< _CharT, _InIter >`
- class `std::money_put< _CharT, _OutIter >`
- class `std::moneypunct< _CharT, _Intl >`
- class `std::num_get< _CharT, _InIter >`
- class `std::num_put< _CharT, _OutIter >`
- class `std::numpunct< _CharT >`
- class `std::time_base`
- class `std::time_get< _CharT, _InIter >`
- class `std::time_put< _CharT, _OutIter >`
- class `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`
- class `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`

Functions

- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn >`
`bool std::do_str_codecvt (const _InChar * __first, const _InChar * __last, _OutStr & __outstr, const _Codecvt & __cvt, _State & __state, size_t & __count, _Fn __fn)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_in (const char * __first, const char * __last, basic_string< _CharT, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt, _State & __state, size_t & __count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_in (const char * __first, const char * __last, basic_string< _CharT, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_in_all (const char * __first, const char * __last, basic_string< _CharT, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_out (const _CharT * __first, const _CharT * __last, basic_string< char, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt, _State & __state, size_t & __count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_out (const _CharT * __first, const _CharT * __last, basic_string< char, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`
`bool std::__str_codecvt_out_all (const _CharT * __first, const _CharT * __last, basic_string< char, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt)`
- `template<typename _Facet >`
`bool std::has_facet (const locale & __loc) throw ()`
- `template<typename _Facet >`
`const _Facet & std::use_facet (const locale & __loc)`

2.48.1 Detailed Description

Classes and functions for internationalization and localization.

2.48.2 Function Documentation

2.48.2.1 `has_facet()`

```
template<typename _Facet >
bool std::has_facet (
    const locale & __loc ) throw ( )
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

<code>_Facet</code>	The facet type to test the presence of.
---------------------	---

Parameters

<code>__loc</code>	The locale to test.
--------------------	---------------------

Returns

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file `locale_classes.tcc`.

2.48.2.2 `use_facet()`

```
template<typename _Facet >
const _Facet & std::use_facet (
    const locale & __loc )
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

Template Parameters

<code>_Facet</code>	The facet type to access.
---------------------	---------------------------

Parameters

<code>__loc</code>	The locale to use.
--------------------	--------------------

Returns

Reference to facet of type `Facet`.

Exceptions

<code>std::bad_cast</code>	if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> .
----------------------------	---

Definition at line 132 of file `locale_classes.tcc`.

2.49 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb >`
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`

- float [__gnu_cxx::conf_hypergf](#) (float __a, float __c, float __x)
- long double [__gnu_cxx::conf_hypergl](#) (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp >
 [__gnu_cxx::__promote_2](#)< _Tpnu, _Tp >::__type [std::cyl_bessel_i](#) (_Tpnu __nu, _Tp __x)
- float [std::cyl_bessel_if](#) (float __nu, float __x)
- long double [std::cyl_bessel_il](#) (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
 [__gnu_cxx::__promote_2](#)< _Tpnu, _Tp >::__type [std::cyl_bessel_j](#) (_Tpnu __nu, _Tp __x)
- float [std::cyl_bessel_jf](#) (float __nu, float __x)
- long double [std::cyl_bessel_jl](#) (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
 [__gnu_cxx::__promote_2](#)< _Tpnu, _Tp >::__type [std::cyl_bessel_k](#) (_Tpnu __nu, _Tp __x)
- float [std::cyl_bessel_kf](#) (float __nu, float __x)
- long double [std::cyl_bessel_kl](#) (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
 [__gnu_cxx::__promote_2](#)< _Tpnu, _Tp >::__type [std::cyl_neumann](#) (_Tpnu __nu, _Tp __x)
- float [std::cyl_neumannf](#) (float __nu, float __x)
- long double [std::cyl_neumannl](#) (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp >
 [__gnu_cxx::__promote_2](#)< _Tp, _Tpp >::__type [std::ellint_1](#) (_Tp __k, _Tpp __phi)
- float [std::ellint_1f](#) (float __k, float __phi)
- long double [std::ellint_1l](#) (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp >
 [__gnu_cxx::__promote_2](#)< _Tp, _Tpp >::__type [std::ellint_2](#) (_Tp __k, _Tpp __phi)
- float [std::ellint_2f](#) (float __k, float __phi)
- long double [std::ellint_2l](#) (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp >
 [__gnu_cxx::__promote_3](#)< _Tp, _Tpn, _Tpp >::__type [std::ellint_3](#) (_Tp __k, _Tpn __nu, _Tpp __phi)
- float [std::ellint_3f](#) (float __k, float __nu, float __phi)
- long double [std::ellint_3l](#) (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
 [__gnu_cxx::__promote](#)< _Tp >::__type [std::expint](#) (_Tp __x)
- float [std::expintf](#) (float __x)
- long double [std::expintl](#) (long double __x)
- template<typename _Tp >
 [__gnu_cxx::__promote](#)< _Tp >::__type [std::hermite](#) (unsigned int __n, _Tp __x)
- float [std::hermitef](#) (unsigned int __n, float __x)
- long double [std::hermitel](#) (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >
 [__gnu_cxx::__promote_4](#)< _Tpa, _Tpb, _Tpc, _Tp >::__type [__gnu_cxx::hyperg](#) (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float [__gnu_cxx::hypergf](#) (float __a, float __b, float __c, float __x)
- long double [__gnu_cxx::hypergl](#) (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
 [__gnu_cxx::__promote](#)< _Tp >::__type [std::laguerre](#) (unsigned int __n, _Tp __x)
- float [std::laguerref](#) (unsigned int __n, float __x)
- long double [std::laguerrel](#) (unsigned int __n, long double __x)
- template<typename _Tp >
 [__gnu_cxx::__promote](#)< _Tp >::__type [std::legendre](#) (unsigned int __l, _Tp __x)
- float [std::legendref](#) (unsigned int __l, float __x)
- long double [std::legendrel](#) (unsigned int __l, long double __x)

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (_Tp __s)`
- `float std::riemann_zetaf (float __s)`
- `long double std::riemann_zetal (long double __s)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)`
- `float std::sph_besself (unsigned int __n, float __x)`
- `long double std::sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double std::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::sph_neumann (unsigned int __n, _Tp __x)`
- `float std::sph_neumannf (unsigned int __n, float __x)`
- `long double std::sph_neumannl (unsigned int __n, long double __x)`

2.49.1 Detailed Description

2.49.2 Mathematical Special Functions

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124 and then added to ISO C++ 2017.

2.49.2.1 Introduction and History

The first significant library upgrade on the road to C++2011, [TR1](#), included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in `<cmath>`.

Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard [IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions](#).

For C++17 these functions were incorporated into the main standard.

2.49.2.2 Contents

The following functions are implemented in namespace `std`:

- `assoc_laguerre` - Associated Laguerre functions
- `assoc_legendre` - Associated Legendre functions
- `beta` - Beta functions
- `comp_ellint_1` - Complete elliptic functions of the first kind
- `comp_ellint_2` - Complete elliptic functions of the second kind
- `comp_ellint_3` - Complete elliptic functions of the third kind

- `cyl_bessel_i` - Regular modified cylindrical Bessel functions
- `cyl_bessel_j` - Cylindrical Bessel functions of the first kind
- `cyl_bessel_k` - Irregular modified cylindrical Bessel functions
- `cyl_neumann` - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- `ellint_1` - Incomplete elliptic functions of the first kind
- `ellint_2` - Incomplete elliptic functions of the second kind
- `ellint_3` - Incomplete elliptic functions of the third kind
- `expint` - The exponential integral
- `hermite` - Hermite polynomials
- `laguerre` - Laguerre functions
- `legendre` - Legendre polynomials
- `riemann_zeta` - The Riemann zeta function
- `sph_bessel` - Spherical Bessel functions
- `sph_legendre` - Spherical Legendre functions
- `sph_neumann` - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- [conf_hyperg](#) - Confluent hypergeometric functions
- [hyperg](#) - Hypergeometric functions

2.49.2.3 Argument Promotion

The arguments supplied to the non-suffixed functions will be promoted according to the following rules:

1. If any argument intended to be floating point is given an integral value That integral value is promoted to double.
2. All floating point arguments are promoted up to the largest floating point precision among them.

2.49.2.4 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (`std::numeric_limits<Tp>::quiet_↔NaN`), the value NaN is returned.

2.49.2.5 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with `std::numeric_limits` and functions thereof.

We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similarly, `long double` should give you more dynamic range and slightly more precision than `double` on many systems.

2.49.2.6 Testing

These functions have been tested against equivalent implementations from the [Gnu Scientific Library](#), [GSL](#) and [Boost](#) and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within 10^{-15} for 64-bit double on linux-x86_64 systems over most of the ranges of validity.

Todo Provide accuracy comparisons on a per-function basis for a small number of targets.

2.49.2.7 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables
 Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55
 Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example <http://people.math.sfu.ca/~cbm/aands/>

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: <http://dlmf.nist.gov/>↔ This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

2.49.3 Function Documentation

2.49.3.1 `airy_ai()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type __gnu_cxx::airy_ai (
    _Tp __x ) [inline]
```

Return the Airy function $Ai(x)$ of real argument x .

Definition at line 1242 of file `specfun.h`.

2.49.3.2 airy_aif()

```
float __gnu_cxx::airy_aif (
    float __x ) [inline]
```

Return the Airy function $Ai(x)$ of float argument x.

Definition at line 1219 of file specfun.h.

2.49.3.3 airy_ail()

```
long double __gnu_cxx::airy_ail (
    long double __x ) [inline]
```

Return the Airy function $Ai(x)$ of long double argument x.

Definition at line 1230 of file specfun.h.

2.49.3.4 airy_bi()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type __gnu_cxx::airy_bi (
    _Tp __x ) [inline]
```

Return the Airy function $Bi(x)$ of real argument x.

Definition at line 1277 of file specfun.h.

2.49.3.5 airy_bif()

```
float __gnu_cxx::airy_bif (
    float __x ) [inline]
```

Return the Airy function $Bi(x)$ of float argument x.

Definition at line 1254 of file specfun.h.

2.49.3.6 airy_bil()

```
long double __gnu_cxx::airy_bil (
    long double __x ) [inline]
```

Return the Airy function $Bi(x)$ of long double argument x.

Definition at line 1265 of file specfun.h.

2.49.3.7 `assoc_laguerre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x ) [inline]
```

Return the associated Laguerre polynomial of nonnegative order n , nonnegative degree m and real argument $x \leftarrow$: $L_n^m(x)$.

The associated Laguerre function of real degree α , $L_n^\alpha(x)$, is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!} {}_1F_1(-n; \alpha+1; x)$$

where $(\alpha)_n$ is the Pochhammer symbol and ${}_1F_1(a; c; x)$ is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree $\alpha = m$ by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

and $x \geq 0$.

See also

`laguerre` for details of the Laguerre function of degree n

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__n</code>	The order of the Laguerre function, <code>__n</code> ≥ 0 .
<code>__m</code>	The degree of the Laguerre function, <code>__m</code> ≥ 0 .
<code>__x</code>	The argument of the Laguerre function, <code>__x</code> ≥ 0 .

Exceptions

<code>std::domain_error</code>	if <code>__x</code> < 0 .
--------------------------------	-----------------------------

Definition at line 252 of file specfun.h.

2.49.3.8 `assoc_laguerref()`

```
float std::assoc_laguerref (
    unsigned int __n,
    unsigned int __m,
    float __x ) [inline]
```

Return the associated Laguerre polynomial of order n , degree m : $L_n^m(x)$ for `float` argument.

See also

`assoc_laguerre` for more details.

Definition at line 206 of file specfun.h.

2.49.3.9 `assoc_laguerrel()`

```
long double std::assoc_laguerrel (
    unsigned int __n,
    unsigned int __m,
    long double __x ) [inline]
```

Return the associated Laguerre polynomial of order n , degree m : $L_n^m(x)$.

See also

`assoc_laguerre` for more details.

Definition at line 216 of file specfun.h.

2.49.3.10 `assoc_legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x ) [inline]
```

Return the associated Legendre function of degree l and order m .

The associated Legendre function is derived from the Legendre function $P_l(x)$ by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

See also

`legendre` for details of the Legendre function of degree l

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__l</code>	The degree <code>__l >= 0</code> .
<code>__m</code>	The order <code>__m <= 1</code> .
<code>__x</code>	The argument, <code>abs (__x) <= 1</code> .

Exceptions

<code>std::domain_error</code>	if <code>abs (__x) > 1</code> .
--------------------------------	------------------------------------

Definition at line 298 of file `specfun.h`.

2.49.3.11 `assoc_legendref()`

```
float std::assoc_legendref (
    unsigned int __l,
    unsigned int __m,
    float __x ) [inline]
```

Return the associated Legendre function of degree `l` and order `m` for `float` argument.

See also

`assoc_legendre` for more details.

Definition at line 267 of file `specfun.h`.

2.49.3.12 `assoc_legendrel()`

```
long double std::assoc_legendrel (
    unsigned int __l,
    unsigned int __m,
    long double __x ) [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

See also

`assoc_legendre` for more details.

Definition at line 276 of file `specfun.h`.

2.49.3.13 beta()

```
template<typename _Tpa , typename _Tpb >
__gnu_cxx::__promote_2<_Tpa, _Tpb>::__type std::beta (
    _Tpa __a,
    _Tpb __b ) [inline]
```

Return the beta function, $B(a, b)$, for real parameters a, b .

The beta function is defined by

$$B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where $a > 0$ and $b > 0$

Template Parameters

<code>_Tpa</code>	The floating-point type of the parameter <code>__a</code> .
<code>_Tpb</code>	The floating-point type of the parameter <code>__b</code> .

Parameters

<code>__a</code>	The first argument of the beta function, <code>__a > 0</code> .
<code>__b</code>	The second argument of the beta function, <code>__b > 0</code> .

Exceptions

<code>std::domain_error</code>	if <code>__a < 0</code> or <code>__b < 0</code> .
--------------------------------	---

Definition at line 343 of file `specfun.h`.

2.49.3.14 betaf()

```
float std::betaf (
    float __a,
    float __b ) [inline]
```

Return the beta function, $B(a, b)$, for `float` parameters a, b .

See also

`beta` for more details.

Definition at line 312 of file `specfun.h`.

2.49.3.15 `betal()`

```
long double std::betal (
    long double __a,
    long double __b ) [inline]
```

Return the beta function, $B(a, b)$, for long double parameters `a`, `b`.

See also

`beta` for more details.

Definition at line 322 of file `specfun.h`.

2.49.3.16 `comp_ellint_1()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::comp_ellint_1 (
    _Tp __k ) [inline]
```

Return the complete elliptic integral of the first kind $K(k)$ for real modulus `k`.

The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

where $F(k, \phi)$ is the incomplete elliptic integral of the first kind and the modulus $|k| \leq 1$.

See also

`ellint_1` for details of the incomplete elliptic function of the first kind.

Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
------------------	---

Parameters

<code>__k</code>	The modulus, $\text{abs}(\text{__k}) \leq 1$
------------------	--

Exceptions

<code>std::domain_error</code>	if $\text{abs}(\text{__k}) > 1$.
--------------------------------	-----------------------------------

Definition at line 391 of file specfun.h.

2.49.3.17 `comp_ellint_1f()`

```
float std::comp_ellint_1f (
    float __k ) [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for `float` modulus `k`.

See also

`comp_ellint_1` for details.

Definition at line 358 of file specfun.h.

2.49.3.18 `comp_ellint_1l()`

```
long double std::comp_ellint_1l (
    long double __k ) [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for `long double` modulus `k`.

See also

`comp_ellint_1` for details.

Definition at line 368 of file specfun.h.

2.49.3.19 `comp_ellint_2()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::comp_ellint_2 (
    _Tp __k ) [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for real modulus `k`.

The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta}$$

where $E(k, \phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| \leq 1$.

See also

`ellint_2` for details of the incomplete elliptic function of the second kind.

Template Parameters

<code>__Tp</code>	The floating-point type of the modulus <code>__k</code> .
-------------------	---

Parameters

<code>__k</code>	The modulus, <code>abs (__k) <= 1</code>
------------------	---

Exceptions

<code>std::domain_error</code>	if <code>abs (__k) > 1</code> .
--------------------------------	------------------------------------

Definition at line 438 of file `specfun.h`.

2.49.3.20 `comp_ellint_2f()`

```
float std::comp_ellint_2f (
    float __k ) [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for `float` modulus `k`.

See also

`comp_ellint_2` for details.

Definition at line 406 of file `specfun.h`.

2.49.3.21 `comp_ellint_2l()`

```
long double std::comp_ellint_2l (
    long double __k ) [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for long double modulus `k`.

See also

`comp_ellint_2` for details.

Definition at line 416 of file `specfun.h`.

2.49.3.22 comp_ellint_3()

```
template<typename _Tp , typename _Tpn >
__gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu ) [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k, \nu) = \Pi(k, \nu, \pi/2)$ for real modulus k .

The complete elliptic integral of the third kind is defined as

$$\Pi(k, \nu) = \Pi(k, \nu, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

where $\Pi(k, \nu, \phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| \leq 1$.

See also

`ellint_3` for details of the incomplete elliptic function of the third kind.

Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpn</code>	The floating-point type of the argument <code>__nu</code> .

Parameters

<code>__k</code>	The modulus, <code>abs (__k) <= 1</code>
<code>__nu</code>	The argument

Exceptions

<code>std::domain_error</code>	if <code>abs (__k) > 1</code> .
--------------------------------	------------------------------------

Definition at line 489 of file `specfun.h`.

2.49.3.23 comp_ellint_3f()

```
float std::comp_ellint_3f (
    float __k,
    float __nu ) [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k, \nu)$ for `float` modulus k .

See also

`comp_ellint_3` for details.

Definition at line 453 of file `specfun.h`.

2.49.3.24 `comp_ellint_3l()`

```
long double std::comp_ellint_3l (
    long double __k,
    long double __nu ) [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k, \nu)$ for `long double` modulus `k`.

See also

`comp_ellint_3` for details.

Definition at line 463 of file `specfun.h`.

2.49.3.25 `conf_hyperg()`

```
template<typename _Tpa , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type __gnu_cxx::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x ) [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a; c; x)$ of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

The confluent hypergeometric function is defined by

$${}_1F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is $(x)_k = (x)(x+1)\dots(x+k-1)$, $(x)_0 = 1$

Parameters

<code>↔ _a</code>	The numeratorial parameter
<code>↔ _c</code>	The denominatorial parameter
<code>↔ _x</code>	The argument

Definition at line 1327 of file `specfun.h`.

2.49.3.26 `conf_hypergf()`

```
float __gnu_cxx::conf_hypergf (
    float __a,
```

```
float __c,
float __x ) [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a; c; x)$ of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

`conf_hyperg` for details.

Definition at line 1295 of file `specfun.h`.

2.49.3.27 `conf_hypergl()`

```
long double __gnu_cxx::conf_hypergl (
    long double __a,
    long double __c,
    long double __x ) [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a; c; x)$ of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

`conf_hyperg` for details.

Definition at line 1306 of file `specfun.h`.

2.49.3.28 `cyl_bessel_i()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for real order ν and argument $x \geq 0$.

The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x >= 0</code>

Exceptions

<code>std::domain_error</code>	if <code>__x < 0</code> .
--------------------------------	------------------------------

Definition at line 535 of file `specfun.h`.

2.49.3.29 `cyl_bessel_if()`

```
float std::cyl_bessel_if (
    float __nu,
    float __x ) [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `float` order ν and argument $x \geq 0$.

See also

`cyl_bessel_i` for details.

Definition at line 504 of file `specfun.h`.

2.49.3.30 `cyl_bessel_il()`

```
long double std::cyl_bessel_il (
    long double __nu,
    long double __x ) [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `long double` order ν and argument $x \geq 0$.

See also

`cyl_bessel_i` for details.

Definition at line 514 of file `specfun.h`.

2.49.3.31 `cyl_bessel_j()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the Bessel function $J_\nu(x)$ of real order ν and argument $x \geq 0$.

The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}$$

Template Parameters

<code>__Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>__Tp</code>	The floating-point type of the argument <code>__x</code> .

Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x >= 0</code>

Exceptions

<code>std::domain_error</code>	if <code>__x < 0</code> .
--------------------------------	------------------------------

Definition at line 581 of file `specfun.h`.

2.49.3.32 `cyl_bessel_jf()`

```
float std::cyl_bessel_jf (  
    float __nu,  
    float __x ) [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `float` order ν and argument $x \geq 0$.

See also

`cyl_bessel_j` for setails.

Definition at line 550 of file `specfun.h`.

2.49.3.33 `cyl_bessel_jl()`

```
long double std::cyl_bessel_jl (  
    long double __nu,  
    long double __x ) [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `long double` order ν and argument $x \geq 0$.

See also

`cyl_bessel_j` for setails.

Definition at line 560 of file `specfun.h`.

2.49.3.34 `cyl_bessel_k()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ of real order ν and argument x .

The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2} \frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}$$

where for integral $\nu = n$ a limit is taken: $\lim_{\nu \rightarrow n}$. For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x >= 0</code>

Exceptions

<code>std::domain_error</code>	if <code>__x < 0</code> .
--------------------------------	------------------------------

Definition at line 633 of file `specfun.h`.

2.49.3.35 `cyl_bessel_kf()`

```
float std::cyl_bessel_kf (
    float __nu,
    float __x ) [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for `float` order ν and argument $x \geq 0$.

See also

`cyl_bessel_k` for details.

Definition at line 596 of file `specfun.h`.

2.49.3.36 cyl_bessel_kl()

```
long double std::cyl_bessel_kl (
    long double __nu,
    long double __x ) [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for long double order ν and argument $x \geq 0$.

See also

`cyl_bessel_k` for setails.

Definition at line 606 of file specfun.h.

2.49.3.37 cyl_neumann()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::cyl_neumann (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

Return the Neumann function $N_\nu(x)$ of real order ν and argument $x \geq 0$.

The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x) \cos \nu\pi - J_{-\nu}(x)}{\sin \nu\pi}$$

where $x \geq 0$ and for integral order $\nu = n$ a limit is taken: $\lim_{\nu \rightarrow n}$.

Template Parameters

<code>_Tpnu</code>	The floating-point type of the order <code>__nu</code> .
<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .

Parameters

<code>__nu</code>	The order
<code>__x</code>	The argument, <code>__x</code> ≥ 0

Exceptions

<code>std::domain_error</code>	if <code>__x</code> < 0 .
--------------------------------	-----------------------------

Definition at line 681 of file specfun.h.

2.49.3.38 cyl_neumannf()

```
float std::cyl_neumannf (
    float __nu,
    float __x ) [inline]
```

Return the Neumann function $N_\nu(x)$ of `float` order ν and argument x .

See also

`cyl_neumann` for setails.

Definition at line 648 of file `specfun.h`.

2.49.3.39 cyl_neumannl()

```
long double std::cyl_neumannl (
    long double __nu,
    long double __x ) [inline]
```

Return the Neumann function $N_\nu(x)$ of `long double` order ν and argument x .

See also

`cyl_neumann` for setails.

Definition at line 658 of file `specfun.h`.

2.49.3.40 ellint_1()

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_1 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind $F(k, \phi)$ for `real` modulus k and angle ϕ .

The incomplete elliptic integral of the first kind is defined as

$$F(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the first kind, $K(k)$.

See also

`comp_ellint_1`.

Template Parameters

<code>__Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>__Tpp</code>	The floating-point type of the angle <code>__phi</code> .

Parameters

<code>__k</code>	The modulus, <code>abs (__k) <= 1</code>
<code>__phi</code>	The integral limit argument in radians

Exceptions

<code>std::domain_error</code>	if <code>abs (__k) > 1</code> .
--------------------------------	------------------------------------

Definition at line 729 of file `specfun.h`.

2.49.3.41 `ellint_1f()`

```
float std::ellint_1f (  
    float __k,  
    float __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind $E(k, \phi)$ for `float` modulus k and angle ϕ .

See also

`ellint_1` for details.

Definition at line 696 of file `specfun.h`.

2.49.3.42 `ellint_1l()`

```
long double std::ellint_1l (  
    long double __k,  
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the first kind $E(k, \phi)$ for `long double` modulus k and angle ϕ .

See also

`ellint_1` for details.

Definition at line 706 of file `specfun.h`.

2.49.3.43 `ellint_2()`

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::ellint_2 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.

The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^\phi \sqrt{1 - k^2 \sin^2 \theta}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the second kind, $E(k)$.

See also

`comp_ellint_2`.

Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

Parameters

<code>__k</code>	The modulus, <code>abs (__k) <= 1</code>
<code>__phi</code>	The integral limit argument in radians

Returns

The elliptic function of the second kind.

Exceptions

<code>std::domain_error</code>	if <code>abs (__k) > 1</code> .
--------------------------------	------------------------------------

Definition at line 777 of file `specfun.h`.

2.49.3.44 `ellint_2f()`

```
float std::ellint_2f (
    float __k,
    float __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$ for `float` argument.

See also

`ellint_2` for details.

Definition at line 744 of file `specfun.h`.

2.49.3.45 `ellint_2l()`

```
long double std::ellint_2l (
    long double __k,
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.

See also

`ellint_2` for details.

Definition at line 754 of file `specfun.h`.

2.49.3.46 `ellint_3()`

```
template<typename _Tp , typename _Tpn , typename _Tpp >
__gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.

The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the third kind, $\Pi(k, \nu)$.

See also

`comp_ellint_3`.

Template Parameters

<code>_Tp</code>	The floating-point type of the modulus <code>__k</code> .
<code>_Tpn</code>	The floating-point type of the argument <code>__nu</code> .
<code>_Tpp</code>	The floating-point type of the angle <code>__phi</code> .

Parameters

<code>__k</code>	The modulus, <code>abs (__k) <= 1</code>
<code>__nu</code>	The second argument
<code>__phi</code>	The integral limit argument in radians

Returns

The elliptic function of the third kind.

Exceptions

<code>std::domain_error</code>	if <code>abs (__k) > 1</code> .
--------------------------------	------------------------------------

Definition at line 830 of file `specfun.h`.

2.49.3.47 `ellint_3f()`

```
float std::ellint_3f (
    float __k,
    float __nu,
    float __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$ for `float` argument.

See also

`ellint_3` for details.

Definition at line 792 of file `specfun.h`.

2.49.3.48 `ellint_3l()`

```
long double std::ellint_3l (
    long double __k,
    long double __nu,
    long double __phi ) [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.

See also

`ellint_3` for details.

Definition at line 802 of file `specfun.h`.

2.49.3.49 expint()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::expint (
    _Tp __x ) [inline]
```

Return the exponential integral $Ei(x)$ for real argument `x`.

The exponential integral is given by

$$Ei(x) = - \int_{-x}^{\infty} \frac{e^t}{t} dt$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__x</code>	The argument of the exponential integral function.
------------------	--

Definition at line 870 of file `specfun.h`.

2.49.3.50 expintf()

```
float std::expintf (
    float __x ) [inline]
```

Return the exponential integral $Ei(x)$ for `float` argument `x`.

See also

`expint` for details.

Definition at line 844 of file `specfun.h`.

2.49.3.51 expintl()

```
long double std::expintl (
    long double __x ) [inline]
```

Return the exponential integral $Ei(x)$ for `long double` argument `x`.

See also

`expint` for details.

Definition at line 854 of file `specfun.h`.

2.49.3.52 hermite()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::hermite (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the Hermite polynomial $H_n(x)$ of order `n` and `real` argument `x`.

The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__n</code>	The order
<code>__x</code>	The argument

Definition at line 918 of file `specfun.h`.

2.49.3.53 hermitef()

```
float std::hermitef (
    unsigned int __n,
    float __x ) [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order `n` and `float` argument `x`.

See also

`hermite` for details.

Definition at line 885 of file `specfun.h`.

2.49.3.54 hermitel()

```
long double std::hermitel (
    unsigned int __n,
    long double __x ) [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order n and `long double` argument x .

See also

`hermite` for details.

Definition at line 895 of file `specfun.h`.

2.49.3.55 hyperg()

```
template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type __gnu_cxx::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x ) [inline]
```

Return the hypergeometric function ${}_2F_1(a, b; c; x)$ of real numeratorial parameters a and b , denominatorial parameter c , and argument x .

The hypergeometric function is defined by

$${}_2F_1(a; c; x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}$$

where the Pochhammer symbol is $(x)_k = (x)(x+1)\dots(x+k-1)$, $(x)_0 = 1$

Parameters

\longleftrightarrow <code>_a</code>	The first numeratorial parameter
\longleftrightarrow <code>_b</code>	The second numeratorial parameter
\longleftrightarrow <code>_c</code>	The denominatorial parameter
\longleftrightarrow <code>_x</code>	The argument

Definition at line 1376 of file `specfun.h`.

2.49.3.56 hypergf()

```
float __gnu_cxx::hypergf (
    float __a,
    float __b,
    float __c,
    float __x ) [inline]
```

Return the hypergeometric function ${}_2F_1(a, b; c; x)$ of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

Definition at line 1343 of file specfun.h.

2.49.3.57 hypergl()

```
long double __gnu_cxx::hypergl (
    long double __a,
    long double __b,
    long double __c,
    long double __x ) [inline]
```

Return the hypergeometric function ${}_2F_1(a, b; c; x)$ of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

Definition at line 1354 of file specfun.h.

2.49.3.58 laguerre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::laguerre (
    unsigned int __n,
    _Tp __x ) [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree n and real argument $x \geq 0$.

The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__n</code>	The nonnegative order
<code>__x</code>	The argument <code>__x >= 0</code>

Exceptions

<code>std::domain_error</code>	if <code>__x < 0</code> .
--------------------------------	------------------------------

Definition at line 962 of file `specfun.h`.

2.49.3.59 `laguerref()`

```
float std::laguerref (
    unsigned int __n,
    float __x ) [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree `n` and `float` argument $x \geq 0$.

See also

`laguerre` for more details.

Definition at line 933 of file `specfun.h`.

2.49.3.60 `laguerrel()`

```
long double std::laguerrel (
    unsigned int __n,
    long double __x ) [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree `n` and `long double` argument $x \geq 0$.

See also

`laguerre` for more details.

Definition at line 943 of file `specfun.h`.

2.49.3.61 `legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::legendre (
    unsigned int __l,
    _Tp __x ) [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and real argument $|x| \leq 0$.

The Legendre function of order l and argument x , $P_l(x)$, is defined by:

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2 - 1)^l$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__l</code>	The degree $l \geq 0$
<code>__x</code>	The argument $\text{abs}(\text{__x}) \leq 1$

Exceptions

<code>std::domain_error</code>	if $\text{abs}(\text{__x}) > 1$
--------------------------------	---------------------------------

Definition at line 1007 of file `specfun.h`.

2.49.3.62 `legendref()`

```
float std::legendref (
    unsigned int __l,
    float __x ) [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and `float` argument $|x| \leq 0$.

See also

`legendre` for more details.

Definition at line 977 of file `specfun.h`.

2.49.3.63 `legendrel()`

```
long double std::legendrel (
    unsigned int __l,
    long double __x ) [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and long double argument $|x| \leq 0$.

See also

`legendre` for more details.

Definition at line 987 of file `specfun.h`.

2.49.3.64 `riemann_zeta()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::riemann_zeta (
    _Tp __s ) [inline]
```

Return the Riemann zeta function $\zeta(s)$ for real argument s .

The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 \leq s \leq 1$$

For $s < 1$ use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin\left(\frac{\pi s}{2}\right) \Gamma(1-s) \zeta(1-s)$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__s</code> .
------------------	--

Parameters

<code>__s</code>	The argument $s \neq 1$
------------------	-------------------------

Definition at line 1058 of file `specfun.h`.

2.49.3.65 riemann_zetaf()

```
float std::riemann_zetaf (
    float __s ) [inline]
```

Return the Riemann zeta function $\zeta(s)$ for `float` argument s .

See also

`riemann_zeta` for more details.

Definition at line 1022 of file `specfun.h`.

2.49.3.66 riemann_zetal()

```
long double std::riemann_zetal (
    long double __s ) [inline]
```

Return the Riemann zeta function $\zeta(s)$ for `long double` argument s .

See also

`riemann_zeta` for more details.

Definition at line 1032 of file `specfun.h`.

2.49.3.67 sph_bessel()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_bessel (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and real argument $x \geq 0$.

The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

\leftrightarrow __n	The integral order n ≥ 0
\leftrightarrow __x	The real argument x ≥ 0

Exceptions

<code>std::domain_error</code>	if __x < 0 .
--------------------------------	--------------

Definition at line 1102 of file specfun.h.

2.49.3.68 sph_besself()

```
float std::sph_besself (
    unsigned int __n,
    float __x ) [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and float argument $x \geq 0$.

See also

sph_bessel for more details.

Definition at line 1073 of file specfun.h.

2.49.3.69 sph_bessell()

```
long double std::sph_bessell (
    unsigned int __n,
    long double __x ) [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and long double argument $x \geq 0$.

See also

sph_bessel for more details.

Definition at line 1083 of file specfun.h.

2.49.3.70 sph_legendre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree l and order m and real angle θ in radians.

The spherical Legendre function is defined by

$$Y_l^m(\theta, \phi) = (-1)^m \left[\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!} \right] P_l^m(\cos \theta) \exp^{im\phi}$$

Template Parameters

<code>_Tp</code>	The floating-point type of the angle <code>__theta</code> .
------------------	---

Parameters

<code>__l</code>	The order <code>__l</code> ≥ 0
<code>__m</code>	The degree <code>__m</code> ≥ 0 and <code>__m</code> \leq <code>__l</code>
<code>__theta</code>	The radian polar angle argument

Definition at line 1149 of file specfun.h.

2.49.3.71 sph_legendref()

```
float std::sph_legendref (
    unsigned int __l,
    unsigned int __m,
    float __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree l and order m and float angle θ in radians.

See also

`sph_legendre` for details.

Definition at line 1117 of file specfun.h.

2.49.3.72 sph_legendrel()

```
long double std::sph_legendrel (
    unsigned int __l,
    unsigned int __m,
    long double __theta ) [inline]
```

Return the spherical Legendre function of nonnegative integral degree l and order m and long double angle θ in radians.

See also

`sph_legendre` for details.

Definition at line 1128 of file `specfun.h`.

2.49.3.73 sph_neumann()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::sph_neumann (
    unsigned int __n,
    _Tp __x ) [inline]
```

Return the spherical Neumann function of integral order $n \geq 0$ and real argument $x \geq 0$.

The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

Template Parameters

<code>_Tp</code>	The floating-point type of the argument <code>__x</code> .
------------------	--

Parameters

<code>__n</code>	The integral order $n \geq 0$
<code>__x</code>	The real argument $__x \geq 0$

Exceptions

<code>std::domain_error</code>	if <code>__x < 0</code> .
--------------------------------	------------------------------

Definition at line 1193 of file `specfun.h`.

2.49.3.74 sph_neumannf()

```
float std::sph_neumannf (
    unsigned int __n,
    float __x ) [inline]
```

Return the spherical Neumann function of integral order $n \geq 0$ and `float` argument $x \geq 0$.

See also

`sph_neumann` for details.

Definition at line 1164 of file `specfun.h`.

2.49.3.75 sph_neumannl()

```
long double std::sph_neumannl (
    unsigned int __n,
    long double __x ) [inline]
```

Return the spherical Neumann function of integral order $n \geq 0$ and `long double` $x \geq 0$.

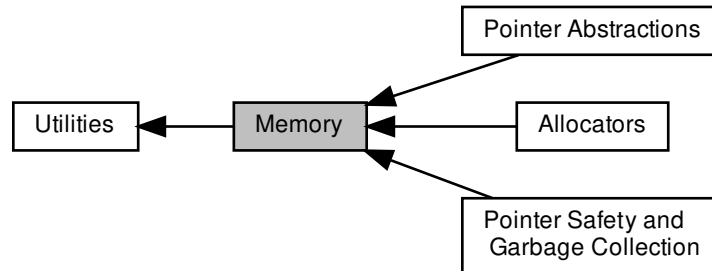
See also

`sph_neumann` for details.

Definition at line 1174 of file `specfun.h`.

2.50 Memory

Collaboration diagram for Memory:



Modules

- [Allocators](#)
- [Pointer Abstractions](#)
- [Pointer Safety and Garbage Collection](#)

Files

- file [memory](#)

Functions

- void * [std::align](#) (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<typename _InputIterator, typename _ForwardIterator >
_ForwardIterator [std::uninitialized_copy](#) (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator >
_ForwardIterator [std::uninitialized_copy_n](#) (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator, typename _Tp >
void [std::uninitialized_fill](#) (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp >
_ForwardIterator [std::uninitialized_fill_n](#) (_ForwardIterator __first, _Size __n, const _Tp &__x)

2.50.1 Detailed Description

Components for memory allocation, deallocation, and management.

2.50.2 Function Documentation

2.50.2.1 align()

```
void* std::align (
    size_t __align,
    size_t __size,
    void *& __ptr,
    size_t & __space ) [inline], [noexcept]
```

Fit aligned storage in buffer.

This function tries to fit `__size` bytes of storage with alignment `__align` into the buffer `__ptr` of size `__space` bytes. If such a buffer fits then `__ptr` is changed to point to the first byte of the aligned storage and `__space` is reduced by the bytes used for alignment.

C++11 20.6.5 [ptr.align]

Parameters

<code>__align</code>	A fundamental or extended alignment value.
<code>__size</code>	Size of the aligned storage required.
<code>__ptr</code>	Pointer to a buffer of <code>__space</code> bytes.
<code>__space</code>	Size of the buffer pointed to by <code>__ptr</code> .

Returns

the updated pointer if the aligned storage fits, otherwise `nullptr`.

Definition at line 123 of file `memory`.

2.50.2.2 uninitialized_copy()

```
template<typename _InputIterator , typename _ForwardIterator >
_FowardIterator std::uninitialized_copy (
    _InputIterator __first,
    _InputIterator __last,
    _ForwardIterator __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`__result + (__first - __last)`

Like `copy()`, but does not require an initialized output range.

Definition at line 125 of file `stl_uninitialized.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms_pu()`.

2.50.2.3 uninitialized_copy_n()

```
template<typename _InputIterator , typename _Size , typename _ForwardIterator >
_FForwardIterator std::uninitialized_copy_n (
    _InputIterator __first,
    _Size __n,
    _ForwardIterator __result ) [inline]
```

Copies the range `[first,first+n)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`__result + __n`

Like `copy_n()`, but does not require an initialized output range.

Definition at line 854 of file `stl_uninitialized.h`.

2.50.2.4 uninitialized_fill()

```
template<typename _ForwardIterator , typename _Tp >
void std::uninitialized_fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __x ) [inline]
```

Copies the value `x` into the range `[first,last)`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__x</code>	The source value.

Returns

Nothing.

Like `fill()`, but does not require an initialized output range.

Definition at line 200 of file `stl_uninitialized.h`.

2.50.2.5 uninitialized_fill_n()

```
template<typename _ForwardIterator , typename _Size , typename _Tp >
_FowardIterator std::uninitialized_fill_n (
    _ForwardIterator __first,
    _Size __n,
    const _Tp & __x ) [inline]
```

Copies the value `x` into the range `[first,first+n)`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of copies to make.
<code>__x</code>	The source value.

Returns

Nothing.

Like `fill_n()`, but does not require an initialized output range.

Definition at line 272 of file `stl_uninitialized.h`.

2.51 Metaprogramming

Collaboration diagram for Metaprogramming:



Classes

- struct `std::__add_pointer_helper< _Tp, bool >`
- struct `std::__detector< _Default, _AlwaysVoid, _Op, _Args >`
- struct `std::__detector< _Default, __void_t< _Op< _Args... > >, _Op, _Args... >`
- struct `std::__is_nullptr_t< _Tp >`
- struct `std::tr2::__reflection_typelist< _Elements >`
- struct `std::tr2::__reflection_typelist< _First, _Rest... >`
- struct `std::tr2::__reflection_typelist<>`
- struct `std::add_const< _Tp >`
- struct `std::add_cv< _Tp >`
- struct `std::add_lvalue_reference< _Tp >`
- struct `std::add_rvalue_reference< _Tp >`
- struct `std::add_volatile< _Tp >`
- struct `std::aligned_storage< _Len, _Align >`
- struct `std::aligned_union< _Len, _Types >`
- struct `std::alignment_of< _Tp >`
- struct `std::tr2::bases< _Tp >`
- struct `std::common_type< _Tp >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- class `std::decay< _Tp >`
- struct `std::tr2::direct_bases< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _Uint >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< typename >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< typename >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_convertible< _From, _To >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`

- struct `std::is_default_constructible< _Tp >`
- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`
- struct `std::is_final< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`
- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_null_pointer< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< typename >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< typename >`
- class `std::reference_wrapper< _Tp >`
- struct `std::remove_all_extents< _Tp >`

- struct `std::remove_const<_Tp>`
- struct `std::remove_cv<_Tp>`
- struct `std::remove_extent<_Tp>`
- struct `std::remove_pointer<_Tp>`
- struct `std::remove_reference<_Tp>`
- struct `std::remove_volatile<_Tp>`
- class `std::result_of<_Signature>`
- struct `std::underlying_type<_Tp>`

Macros

- `#define __cpp_lib_is_final`
- `#define __cpp_lib_is_null_pointer`
- `#define __cpp_lib_result_of_sfinae`
- `#define __cpp_lib_transformation_trait_aliases`
- `#define __cpp_lib_void_t`

Typedefs

- `template<bool __v>`
using `std::__bool_constant` = `integral_constant< bool, __v >`
- `template<typename _Tp>`
using `std::__decay_and_strip` = `__strip_reference_wrapper< __decay_t<_Tp> >`
- `template<typename _Tp>`
using `std::__decay_t` = `typename decay<_Tp>::type`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`
using `std::__detected_or` = `__detector<_Default, void, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`
using `std::__detected_or_t` = `typename __detected_or<_Default, _Op, _Args... >::type`
- `template<bool _Cond, typename _Tp = void>`
using `std::__enable_if_t` = `typename enable_if<_Cond, _Tp>::type`
- `template<typename _ToElementType, typename _FromElementType>`
using `std::__is_array_convertible` = `is_convertible<_FromElementType(*)[], _ToElementType(*)[]>`
- `template<typename _Tp, typename... _Args>`
using `std::__is_nothrow_constructible_impl` = `__is_nt_constructible_impl< __is_constructible(_Tp, _Args...), _Tp, _Args... >`
- `template<typename _Tp, typename... _Types>`
using `std::__is_one_of` = `__or_< is_same<_Tp, _Types>... >`
- `template<typename _Tp>`
using `std::__is_signed_integer` = `__is_one_of< __remove_cv_t<_Tp>, signed char, signed short, signed int, signed long, signed long long >`
- `template<typename _Tp>`
using `std::__is_standard_integer` = `__or_< __is_signed_integer<_Tp>, __is_unsigned_integer<_Tp> >`
- `template<typename _Tp>`
using `std::__is_unsigned_integer` = `__is_one_of< __remove_cv_t<_Tp>, unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long >`
- `template<typename _Tp>`
using `std::__remove_cv_t` = `typename remove_cv<_Tp>::type`
- `template<typename _Tp>`
using `std::__remove_cvref_t` = `typename remove_cv<typename remove_reference<_Tp>::type>::type`

- `template<typename _Tp >`
`using std::__type_identity_t = typename __type_identity< _Tp >::type`
- `template<typename... >`
`using std::__void_t = void`
- `template<typename... _Cond>`
`using std::Require = __enable_if_t< __and< _Cond... >::value >`
- `template<typename _Tp >`
`using std::add_const_t = typename add_const< _Tp >::type`
- `template<typename _Tp >`
`using std::add_cv_t = typename add_cv< _Tp >::type`
- `template<typename _Tp >`
`using std::add_lvalue_reference_t = typename add_lvalue_reference< _Tp >::type`
- `template<typename _Tp >`
`using std::add_pointer_t = typename add_pointer< _Tp >::type`
- `template<typename _Tp >`
`using std::add_rvalue_reference_t = typename add_rvalue_reference< _Tp >::type`
- `template<typename _Tp >`
`using std::add_volatile_t = typename add_volatile< _Tp >::type`
- `template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa< _Len >::type)>`
`using std::aligned_storage_t = typename aligned_storage< _Len, _Align >::type`
- `template<size_t _Len, typename... _Types>`
`using std::aligned_union_t = typename aligned_union< _Len, _Types... >::type`
- `template<typename... _Tp>`
`using std::common_type_t = typename common_type< _Tp... >::type`
- `template<bool _Cond, typename _Iftrue, typename _Iffalse >`
`using std::conditional_t = typename conditional< _Cond, _Iftrue, _Iffalse >::type`
- `template<typename _Tp >`
`using std::decay_t = typename decay< _Tp >::type`
- `template<bool _Cond, typename _Tp = void>`
`using std::enable_if_t = typename enable_if< _Cond, _Tp >::type`
- `typedef integral_constant< bool, false > std::false_type`
- `template<typename _Tp >`
`using std::make_signed_t = typename make_signed< _Tp >::type`
- `template<typename _Tp >`
`using std::make_unsigned_t = typename make_unsigned< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_all_extents_t = typename remove_all_extents< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_const_t = typename remove_const< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_cv_t = typename remove_cv< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_extent_t = typename remove_extent< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_pointer_t = typename remove_pointer< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_reference_t = typename remove_reference< _Tp >::type`
- `template<typename _Tp >`
`using std::remove_volatile_t = typename remove_volatile< _Tp >::type`
- `template<typename _Tp >`
`using std::result_of_t = typename result_of< _Tp >::type`
- `typedef integral_constant< bool, true > std::true_type`

- `template<typename _Tp >`
`using std::underlying_type_t = typename underlying_type< _Tp >::type`
- `template<typename... >`
`using std::void_t = void`

Functions

- `template<typename _Tp, size_t = sizeof(_Tp)>`
`constexpr true_type std::__is_complete_or_unbounded (__type_identity< _Tp >)`
- `template<typename _TypeIdentity, typename _NestedType = typename _TypeIdentity::type>`
`constexpr __or_< is_reference< _NestedType >, is_function< _NestedType >, is_void< _NestedType >, __↵`
`is_array_unknown_bounds< _NestedType > >::type std::__is_complete_or_unbounded (_TypeIdentity)`
- `template<typename _Tp >`
`std::__is_nullptr_t is_null_pointer std::_GLIBCXX_DEPRECATED_SUGGEST ("std::is_null_pointer")`
- `template<typename _Tp >`
`auto std::declval () noexcept -> decltype(__declval< _Tp > (0))`

Variables

- [std::is_reference](#) [std::_GLIBCXX_DEPRECATED_SUGGEST](#)
- static const size_t [std::aligned_union](#)< _Len, _Types >::alignment_value
- static constexpr _Tp [std::integral_constant](#)< _Tp, __v >::value

2.51.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

2.51.2 Typedef Documentation

2.51.2.1 `add_const_t`

```
template<typename _Tp >
using std::add\_const\_t = typedef typename add\_const<_Tp>::type
```

Alias template for `add_const`.

Definition at line 1578 of file `type_traits`.

2.51.2.2 add_cv_t

```
template<typename _Tp >  
using std::add_cv_t = typedef typename add_cv<_Tp>::type
```

Alias template for add_cv.

Definition at line 1586 of file type_traits.

2.51.2.3 add_lvalue_reference_t

```
template<typename _Tp >  
using std::add_lvalue_reference_t = typedef typename add_lvalue_reference<_Tp>::type
```

Alias template for add_lvalue_reference.

Definition at line 1639 of file type_traits.

2.51.2.4 add_pointer_t

```
template<typename _Tp >  
using std::add_pointer_t = typedef typename add_pointer<_Tp>::type
```

Alias template for add_pointer.

Definition at line 2044 of file type_traits.

2.51.2.5 add_rvalue_reference_t

```
template<typename _Tp >  
using std::add_rvalue_reference_t = typedef typename add_rvalue_reference<_Tp>::type
```

Alias template for add_rvalue_reference.

Definition at line 1643 of file type_traits.

2.51.2.6 add_volatile_t

```
template<typename _Tp >  
using std::add_volatile_t = typedef typename add_volatile<_Tp>::type
```

Alias template for add_volatile.

Definition at line 1582 of file type_traits.

2.51.2.7 aligned_storage_t

```
template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>
using std::aligned_storage_t = typedef typename aligned_storage<_Len, _Align>::type
```

Alias template for aligned_storage.

Definition at line 2543 of file type_traits.

2.51.2.8 common_type_t

```
template<typename... _Tp>
using std::common_type_t = typedef typename common_type<_Tp...>::type
```

Alias template for common_type.

Definition at line 2562 of file type_traits.

2.51.2.9 conditional_t

```
template<bool _Cond, typename _Iftrue , typename _Iffalse >
using std::conditional_t = typedef typename conditional<_Cond, _Iftrue, _Iffalse>::type
```

Alias template for conditional.

Definition at line 2558 of file type_traits.

2.51.2.10 decay_t

```
template<typename _Tp >
using std::decay_t = typedef typename decay<_Tp>::type
```

Alias template for decay.

Definition at line 2550 of file type_traits.

2.51.2.11 enable_if_t

```
template<bool _Cond, typename _Tp = void>
using std::enable_if_t = typedef typename enable_if<_Cond, _Tp>::type
```

Alias template for enable_if.

Definition at line 2554 of file type_traits.

2.51.2.12 false_type

```
typedef integral_constant<bool, false> std::false_type
```

The type used as a compile-time boolean with false value.

Definition at line 78 of file type_traits.

2.51.2.13 make_signed_t

```
template<typename _Tp >  
using std::make_signed_t = typedef typename make_signed<_Tp>::type
```

Alias template for make_signed.

Definition at line 1961 of file type_traits.

2.51.2.14 make_unsigned_t

```
template<typename _Tp >  
using std::make_unsigned_t = typedef typename make_unsigned<_Tp>::type
```

Alias template for make_unsigned.

Definition at line 1965 of file type_traits.

2.51.2.15 remove_all_extents_t

```
template<typename _Tp >  
using std::remove_all_extents_t = typedef typename remove_all_extents<_Tp>::type
```

Alias template for remove_all_extents.

Definition at line 2003 of file type_traits.

2.51.2.16 remove_const_t

```
template<typename _Tp >  
using std::remove_const_t = typedef typename remove_const<_Tp>::type
```

Alias template for remove_const.

Definition at line 1566 of file type_traits.

2.51.2.17 remove_cv_t

```
template<typename _Tp >
using std::remove_cv_t = typedef typename remove_cv<_Tp>::type
```

Alias template for remove_cv.

Definition at line 1574 of file type_traits.

2.51.2.18 remove_extent_t

```
template<typename _Tp >
using std::remove_extent_t = typedef typename remove_extent<_Tp>::type
```

Alias template for remove_extent.

Definition at line 1999 of file type_traits.

2.51.2.19 remove_pointer_t

```
template<typename _Tp >
using std::remove_pointer_t = typedef typename remove_pointer<_Tp>::type
```

Alias template for remove_pointer.

Definition at line 2040 of file type_traits.

2.51.2.20 remove_reference_t

```
template<typename _Tp >
using std::remove_reference_t = typedef typename remove_reference<_Tp>::type
```

Alias template for remove_reference.

Definition at line 1635 of file type_traits.

2.51.2.21 remove_volatile_t

```
template<typename _Tp >
using std::remove_volatile_t = typedef typename remove_volatile<_Tp>::type
```

Alias template for remove_volatile.

Definition at line 1570 of file type_traits.

2.51.2.22 result_of_t

```
template<typename _Tp >  
using std::result_of_t = typedef typename result_of<_Tp>::type
```

Alias template for result_of.

Definition at line 2570 of file type_traits.

2.51.2.23 true_type

```
typedef integral_constant<bool, true> std::true_type
```

The type used as a compile-time boolean with true value.

Definition at line 75 of file type_traits.

2.51.2.24 underlying_type_t

```
template<typename _Tp >  
using std::underlying_type_t = typedef typename underlying_type<_Tp>::type
```

Alias template for underlying_type.

Definition at line 2566 of file type_traits.

2.51.2.25 void_t

```
template<typename... >  
using std::void_t = typedef void
```

A metafunction that always yields void, used for detecting valid types.

Definition at line 2576 of file type_traits.

2.51.3 Variable Documentation

2.51.3.1 alignment_value

```
template<size_t _Len, typename... _Types>  
const size_t std::aligned_union< _Len, _Types >::alignment_value [static]
```

The value of the strictest alignment of _Types.

Definition at line 2117 of file type_traits.

2.52 Mutating

Collaboration diagram for Mutating:



Functions

- `template<typename _II, typename _OI >`
`constexpr _OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`constexpr _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↵ Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`constexpr _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`
`constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Generator >`
`constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`
`constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II, typename _OI >`
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`
`constexpr _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator, typename _Predicate >`
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`
`constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator ↵ __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`constexpr _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate ↵ __pred)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber↵ Generator &&__rand)`

- `template<typename _ForwardIterator, typename _Tp >`
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`
`constexpr _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`
`constexpr _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`
`constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`
`constexpr _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`
`constexpr void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`
`constexpr _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::V2::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _OutputIterator >`
`constexpr _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _ForwardIterator, typename _Predicate >`
`_ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`
`constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`
`constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`

2.52.1 Detailed Description

2.52.2 Function Documentation

2.52.2.1 `copy()`

```
template<typename _II , typename _OI >
constexpr _OI std::copy (
    _II __first,
    _II __last,
    _OI __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

`result + (last - first)`

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within `[first,last)`; the `copy_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within `[first,last)`.

Definition at line 560 of file `stl_algobase.h`.

2.52.2.2 `copy_backward()`

```
template<typename _BI1 , typename _BI2 >
constexpr _BI2 std::copy_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result ) [inline]
```

Copies the range `[first,last)` into `result`.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

result - (last - first)

The function has the same effect as `copy`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `copy` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 797 of file `stl_algobase.h`.

Referenced by `std::deque<_StateSeqT>::_M_reallocate_map()`.

2.52.2.3 copy_if()

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
constexpr _OutputIterator std::copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred )
```

Copy the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns true to the range beginning at `__result`.

`copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 688 of file `stl_algo.h`.

2.52.2.4 `copy_n()`

```
template<typename _InputIterator , typename _Size , typename _OutputIterator >
constexpr _OutputIterator std::copy_n (
    _InputIterator __first,
    _Size __n,
    _OutputIterator __result ) [inline]
```

Copies the range `[first,first+n)` into `[result,result+n)`.

Parameters

<code>__first</code>	An input iterator.
<code>__n</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

`result+n`.

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 768 of file `stl_algo.h`.

2.52.2.5 `fill()`

```
template<typename _ForwardIterator , typename _Tp >
constexpr void std::fill (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value ) [inline]
```

Fills the range `[first,last)` with copies of `value`.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

Nothing.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

Definition at line 937 of file `stl_algobase.h`.

2.52.2.6 `fill_n()`

```
template<typename _OI , typename _Size , typename _Tp >
constexpr _OI std::fill_n (
    _OI __first,
    _Size __n,
    const _Tp & __value ) [inline]
```

Fills the range `[first,first+n)` with copies of `value`.

Parameters

<code>__first</code>	An output iterator.
<code>__n</code>	The count of copies to perform.
<code>__value</code>	A reference-to-const of arbitrary type.

Returns

The iterator at `first+n`.

This function fills a range with copies of the same value. For char types filling contiguous areas of memory, this becomes an inline call to `memset` or `wmemset`.

If `__n` is negative, the function does nothing.

Definition at line 1089 of file `stl_algobase.h`.

2.52.2.7 `generate()`

```
template<typename _ForwardIterator , typename _Generator >
constexpr void std::generate (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Generator __gen )
```

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

`generate()` returns no value.

Performs the assignment `*i = __gen()` for each `i` in the range `[__first,__last)`.

Definition at line 4446 of file `stl_algo.h`.

2.52.2.8 `generate_n()`

```
template<typename _OutputIterator , typename _Size , typename _Generator >
constexpr _OutputIterator std::generate_n (
    _OutputIterator __first,
    _Size __n,
    _Generator __gen )
```

Assign the result of a function object to each value in a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__n</code>	The length of the sequence.
<code>__gen</code>	A function object taking no arguments and returning <code>std::iterator_traits<_ForwardIterator>::value_type</code>

Returns

The end of the sequence, `__first+__n`

Performs the assignment `*i = __gen()` for each `i` in the range `[__first,__first+__n)`.

If `__n` is negative, the function does nothing and returns `__first`.

Definition at line 4480 of file `stl_algo.h`.

2.52.2.9 `is_partitioned()`

```
template<typename _InputIterator , typename _Predicate >
constexpr bool std::is_partitioned (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks whether the sequence is partitioned.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the range `[__first,__last)` is partitioned by `__pred`, i.e. if all elements that satisfy `__pred` appear before those that do not.

Definition at line 530 of file `stl_algo.h`.

References `std::find_if_not()`, and `std::none_of()`.

2.52.2.10 `iter_swap()`

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr void std::iter_swap (
    _ForwardIterator1 __a,
    _ForwardIterator2 __b ) [inline]
```

Swaps the contents of two iterators.

Parameters

<code>↔ _a</code>	An iterator.
<code>↔ _b</code>	Another iterator.

Returns

Nothing.

This function swaps the values pointed to by two iterators, not the iterators themselves.

Definition at line 152 of file `stl_algobase.h`.

Referenced by `std::__merge_without_buffer()`, `std::__move_median_to_first()`, `std::__partition()`, `std::__reverse()`, `std::__V2::__rotate()`, and `std::__unguarded_partition()`.

2.52.2.11 move()

```
template<typename _II , typename _OI >
constexpr _OI std::move (
    _II __first,
    _II __last,
    _OI __result ) [inline]
```

Moves the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

result + (last - first)

This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling). Result may not be contained within [first,last); the `move_backward` function should be used instead.

Note that the end of the output range is permitted to be contained within [first,last).

Definition at line 593 of file `stl_algobase.h`.

2.52.2.12 move_backward()

```
template<typename _BI1 , typename _BI2 >
constexpr _BI2 std::move_backward (
    _BI1 __first,
    _BI1 __last,
    _BI2 __result ) [inline]
```

Moves the range [first,last) into result.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	A bidirectional iterator.

Returns

result - (last - first)

The function has the same effect as `move`, but starts at the end of the range and works its way to the start, returning the start of the result. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Result may not be in the range `(first,last]`. Use `move` instead. Note that the start of the output range may overlap `[first,last)`.

Definition at line 833 of file `stl_algobase.h`.

2.52.2.13 partition()

```
template<typename _ForwardIterator , typename _Predicate >
constexpr _ForwardIterator std::partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[__first,middle)` and false for each `i` in the range `[middle,__last)`.

`__pred` must not modify its operand. `partition()` does not preserve the relative ordering of elements in each group, use `stable_partition()` if this is needed.

Definition at line 4668 of file `stl_algo.h`.

2.52.2.14 partition_copy()

```
template<typename _InputIterator , typename _OutputIterator1 , typename _OutputIterator2 , typename
_Predicate >
constexpr pair<_OutputIterator1, _OutputIterator2> std::partition_copy (
    _InputIterator __first,
```

```

__InputIterator __last,
__OutputIterator1 __out_true,
__OutputIterator2 __out_false,
_Predicate __pred )

```

Copy the elements of a sequence to separate output sequences depending on the truth value of a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__out_true</code>	An output iterator.
<code>__out_false</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

A pair designating the ends of the resulting sequences.

Copies each element in the range `[__first, __last)` for which `__pred` returns true to the range beginning at `__out_true` and each element for which `__pred` returns false to `__out_false`.

Definition at line 805 of file `stl_algo.h`.

2.52.2.15 `partition_point()`

```

template<typename _ForwardIterator , typename _Predicate >
constexpr _ForwardIterator std::partition_point (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred )

```

Find the partition point of a partitioned range.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__pred</code>	A predicate.

Returns

An iterator `mid` such that `all_of(__first, mid, __pred)` and `none_of(mid, __last, __pred)` are both true.

Definition at line 552 of file `stl_algo.h`.

2.52.2.16 `random_shuffle()`

```
template<typename _RandomAccessIterator , typename _RandomNumberGenerator >
void std::random_shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _RandomNumberGenerator && __rand )
```

Shuffle the elements of a sequence using a random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__rand</code>	The RNG functor or function.

Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__rand` to provide a random distribution. Calling `__rand(N)` for a positive integer `N` should return a randomly chosen integer from the range `[0,N)`.

Definition at line 4627 of file `stl_algo.h`.

2.52.2.17 `remove()`

```
template<typename _ForwardIterator , typename _Tp >
constexpr _ForwardIterator std::remove (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __value ) [inline]
```

Remove elements from a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

All elements equal to `__value` are removed from the range `[__first,__last)`.

`remove()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 875 of file `stl_algo.h`.

2.52.2.18 `remove_copy()`

```
template<typename _InputIterator , typename _OutputIterator , typename _Tp >
constexpr _OutputIterator std::remove_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    const _Tp & __value ) [inline]
```

Copy a sequence, removing elements of a given value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__value</code>	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` not equal to `__value` to the range beginning at `__result`. `remove_copy()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 619 of file `stl_algo.h`.

2.52.2.19 `remove_copy_if()`

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate >
constexpr _OutputIterator std::remove_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred ) [inline]
```

Copy a sequence, removing elements for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` for which `__pred` returns false to the range beginning at `__result`.

`remove_copy_if()` is stable, so the relative order of elements that are copied is unchanged.

Definition at line 653 of file `stl_algo.h`.

2.52.2.20 `remove_if()`

```
template<typename _ForwardIterator , typename _Predicate >
constexpr _ForwardIterator std::remove_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Remove elements from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.

Returns

An iterator designating the end of the resulting sequence.

All elements for which `__pred` returns true are removed from the range `[__first,__last)`.

`remove_if()` is stable, so the relative order of elements that are not removed is unchanged.

Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 909 of file `stl_algo.h`.

2.52.2.21 replace()

```
template<typename _ForwardIterator , typename _Tp >
constexpr void std::replace (
    _ForwardIterator __first,
    _ForwardIterator __last,
    const _Tp & __old_value,
    const _Tp & __new_value )
```

Replace each occurrence of one value in a sequence with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__old_value</code>	The value to be replaced.
<code>__new_value</code>	The replacement value.

Returns

`replace()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `*i == __old_value` then the assignment `*i = __new_value` is performed.

Definition at line 4380 of file `stl_algo.h`.

2.52.2.22 replace_copy_if()

```
template<typename _InputIterator , typename _OutputIterator , typename _Predicate , typename _Tp
>
constexpr _OutputIterator std::replace_copy_if (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _Predicate __pred,
    const _Tp & __new_value ) [inline]
```

Copy a sequence, replacing each value for which a predicate returns true with another value.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

The end of the output sequence, `__result+(__last-__first)`.

Copies each element in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` replacing elements for which `__pred` returns true with `__new_value`.

Definition at line 3200 of file `stl_algo.h`.

2.52.2.23 `replace_if()`

```
template<typename _ForwardIterator, typename _Predicate, typename _Tp >
constexpr void std::replace_if (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred,
    const _Tp & __new_value )
```

Replace each value in a sequence for which a predicate returns true with another value.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate.
<code>__new_value</code>	The replacement value.

Returns

`replace_if()` returns no value.

For each iterator `i` in the range `[__first,__last)` if `__pred(*i)` is true then the assignment `*i = __new_value` is performed.

Definition at line 4413 of file `stl_algo.h`.

2.52.2.24 `reverse()`

```
template<typename _BidirectionalIterator >
constexpr void std::reverse (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Reverse a sequence.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.

Returns

`reverse()` returns no value.

Reverses the order of the elements in the range `[__first,__last)`, so that the first element becomes the last etc. For every `i` such that $0 \leq i < (_\text{last} - _\text{first})/2$, `reverse()` swaps `*(__first+i)` and `*(__last-(i+1))`

Definition at line 1170 of file `stl_algo.h`.

2.52.2.25 reverse_copy()

```
template<typename _BidirectionalIterator , typename _OutputIterator >
constexpr _OutputIterator std::reverse_copy (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _OutputIterator __result )
```

Copy a sequence, reversing its elements.

Parameters

<code>__first</code>	A bidirectional iterator.
<code>__last</code>	A bidirectional iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements in the range `[__first,__last)` to the range `[__result,__result+(__last-__first))` such that the order of the elements is reversed. For every `i` such that $0 \leq i < (_\text{last} - _\text{first})$, `reverse_copy()` performs the assignment `*(__result+(__last-__first)-1-i) = *(__first+i)`. The ranges `[__first,__last)` and `[__result,__result+(__last-__first))` must not overlap.

Definition at line 1198 of file `stl_algo.h`.

2.52.2.26 rotate()

```
template<typename _ForwardIterator >
constexpr _ForwardIterator std::_V2::rotate (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last ) [inline]
```

Rotate the elements of a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

`first + (last - middle).`

Rotates the elements of the range `[__first,__last)` by `(__middle - __first)` positions so that the element at `__middle` is moved to `__first`, the element at `__middle+1` is moved to `__first+1` and so on for each element in the range `[__first,__last)`.

This effectively swaps the ranges `[__first,__middle)` and `[__middle,__last)`.

Performs `*(__first+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0,__last-__first)`.

Definition at line 1430 of file `stl_algo.h`.

2.52.2.27 rotate_copy()

```
template<typename _ForwardIterator , typename _OutputIterator >
constexpr _OutputIterator std::rotate_copy (
    _ForwardIterator __first,
    _ForwardIterator __middle,
    _ForwardIterator __last,
    _OutputIterator __result ) [inline]
```

Copy a sequence, rotating its elements.

Parameters

<code>__first</code>	A forward iterator.
<code>__middle</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies the elements of the range `[__first,__last)` to the range beginning at

Returns

, rotating the copied elements by `(__middle-__first)` positions so that the element at `__middle` is moved to `↵ __result`, the element at `__middle+1` is moved to `__result+1` and so on for each element in the range `[__first,__last)`.

Performs `*(__result+(n+(__last-__middle))%(__last-__first))=*(__first+n)` for each `n` in the range `[0,__last-__first)`.

Definition at line 1468 of file `stl_algo.h`.

2.52.2.28 shuffle()

```
template<typename _RandomAccessIterator , typename _UniformRandomNumberGenerator >
void std::shuffle (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _UniformRandomNumberGenerator && __g )
```

Shuffle the elements of a sequence using a uniform random number generator.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__g</code>	A <code>UniformRandomNumberGenerator</code> (26.5.1.3).

Returns

Nothing.

Reorders the elements in the range `[__first,__last)` using `__g` to provide random numbers.

Definition at line 3753 of file `stl_algo.h`.

2.52.2.29 stable_partition()

```
template<typename _ForwardIterator , typename _Predicate >
_FowardIterator std::stable_partition (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Predicate __pred ) [inline]
```

Move elements for which a predicate is true to the beginning of a sequence, preserving relative ordering.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__pred</code>	A predicate functor.

Returns

An iterator `middle` such that `__pred(i)` is true for each iterator `i` in the range `[first,middle)` and false for each `i` in the range `[middle,last)`.

Performs the same function as `partition()` with the additional guarantee that the relative ordering of elements in each group is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__pred(x) == __pred(y)` will have the same relative ordering after calling `stable_partition()`.

Definition at line 1649 of file `stl_algo.h`.

2.52.2.30 `swap_ranges()`

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr _ForwardIterator2 std::swap_ranges (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2 )
```

Swap the elements of two sequences.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.

Returns

An iterator equal to `first2+(last1-first1)`.

Swaps each element in the range `[first1,last1)` with the corresponding element in the range `[first2,(last1-first1))`. The ranges must not overlap.

Definition at line 201 of file `stl_algobase.h`.

2.52.2.31 transform() [1/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _UnaryOperation >
constexpr _OutputIterator std::transform (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _UnaryOperation __unary_op )
```

Perform an operation on a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__unary_op</code>	A unary operator.

Returns

An output iterator equal to `__result+(__last-__first)`.

Applies the operator to each element in the input range and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=unary_op(*(__first+N))` for each `N` in the range `[0,__last-__first)`.

`unary_op` must not alter its argument.

Definition at line 4309 of file `stl_algo.h`.

2.52.2.32 transform() [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_BinaryOperation >
constexpr _OutputIterator std::transform (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _OutputIterator __result,
    _BinaryOperation __binary_op )
```

Perform an operation on corresponding elements of two sequences.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_op</code>	A binary operator.

Returns

An output iterator equal to `result+(last-first)`.

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N)=__binary_op(*(__first1+N),*(__first2+N))` for each `N` in the range `[0,__last1-__first1)`.

`binary_op` must not alter either of its arguments.

Definition at line 4347 of file `stl_algo.h`.

2.52.2.33 unique() [1/2]

```
template<typename _ForwardIterator >
constexpr _ForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Remove consecutive duplicate values from a sequence.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values that compare equal. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 978 of file `stl_algo.h`.

2.52.2.34 unique() [2/2]

```
template<typename _ForwardIterator , typename _BinaryPredicate >
constexpr _ForwardIterator std::unique (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred ) [inline]
```

Remove consecutive values from a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

Removes all but the first element from each group of consecutive values for which `__binary_pred` returns true. `unique()` is stable, so the relative order of elements that are not removed is unchanged. Elements between the end of the resulting sequence and `__last` are still present, but their value is unspecified.

Definition at line 1009 of file `stl_algo.h`.

2.52.2.35 `unique_copy()` [1/2]

```
template<typename _InputIterator , typename _OutputIterator >
constexpr _OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result ) [inline]
```

Copy a sequence, removing consecutive duplicate values.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements that compare equal. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 538. 241 again: Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4518 of file `stl_algo.h`.

2.52.2.36 `unique_copy()` [2/2]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
constexpr _OutputIterator std::unique_copy (
    _InputIterator __first,
    _InputIterator __last,
    _OutputIterator __result,
    _BinaryPredicate __binary_pred ) [inline]
```

Copy a sequence, removing consecutive values using a predicate.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__result</code>	An output iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

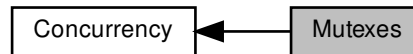
Copies each element in the range `[__first,__last)` to the range beginning at `__result`, except that only the first element is copied from groups of consecutive elements for which `__binary_pred` returns true. `unique_copy()` is stable, so the relative order of elements that are copied is unchanged.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 241. Does `unique_copy()` require CopyConstructible and Assignable?

Definition at line 4560 of file `stl_algo.h`.

2.53 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct [std::adopt_lock_t](#)
- struct [std::defer_lock_t](#)
- class [std::lock_guard< _Mutex >](#)
- class [std::mutex](#)
- struct [std::once_flag](#)
- class [std::recursive_mutex](#)
- class [std::recursive_timed_mutex](#)
- class [std::shared_lock< _Mutex >](#)
- class [std::shared_timed_mutex](#)
- class [std::timed_mutex](#)
- struct [std::try_to_lock_t](#)
- class [std::unique_lock< _Mutex >](#)

Macros

- `#define __cpp_lib_shared_timed_mutex`

Functions

- `template<typename _Callable, typename... _Args>`
`void std::call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)`
- `template<typename _L1, typename _L2, typename... _L3>`
`void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`
`int std::try_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &... __l3)`

Variables

- `constexpr adopt_lock_t std::adopt_lock`
- `constexpr defer_lock_t std::defer_lock`
- `constexpr try_to_lock_t std::try_to_lock`

2.53.1 Detailed Description

Classes for mutex support.

2.53.2 Function Documentation

2.53.2.1 `call_once()`

```
template<typename _Callable , typename... _Args>
void std::call_once (
    once_flag & __once,
    _Callable && __f,
    _Args &&... __args )
```

Invoke a callable and synchronize with other calls using the same flag.

Definition at line 712 of file mutex.

2.53.2.2 `lock()`

```
template<typename _L1 , typename _L2 , typename... _L3>
void std::lock (
    _L1 & __l1,
    _L2 & __l2,
    _L3 &... __l3 )
```

Generic lock.

Parameters

\leftarrow _l1	Meets Lockable requirements (try_lock() may throw).
\leftarrow _l2	Meets Lockable requirements (try_lock() may throw).
\leftarrow _l3	Meets Lockable requirements (try_lock() may throw).

Exceptions

<i>An</i>	exception thrown by an argument's lock() or try_lock() member.
-----------	--

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to `lock()`, `try_lock()` and `unlock()`. If the call exits via an exception any locks that were obtained will be released.




Definition at line 589 of file `mutex`.

2.53.2.3 try_lock()

```
template<typename _Lock1 , typename _Lock2 , typename... _Lock3>
int std::try_lock (
    _Lock1 & __l1,
    _Lock2 & __l2,
    _Lock3 &... __l3 )
```

Generic `try_lock`.

Parameters

 <i>__l1</i>	Meets Lockable requirements (<code>try_lock()</code> may throw).
 <i>__l2</i>	Meets Lockable requirements (<code>try_lock()</code> may throw).
 <i>__l3</i>	Meets Lockable requirements (<code>try_lock()</code> may throw).

Returns

Returns -1 if all `try_lock()` calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls `try_lock()` on each argument.

Definition at line 568 of file `mutex`.

2.53.3 Variable Documentation

2.53.3.1 adopt_lock

```
constexpr adopt_lock_t std::adopt_lock [inline]
```

Tag used to make a scoped lock take ownership of a locked mutex.

Definition at line 145 of file std_mutex.h.

2.53.3.2 defer_lock

```
constexpr defer_lock_t std::defer_lock [inline]
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

Definition at line 139 of file std_mutex.h.

2.53.3.3 try_to_lock

```
constexpr try_to_lock_t std::try_to_lock [inline]
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

Definition at line 142 of file std_mutex.h.

2.54 Negators

Collaboration diagram for Negators:



Classes

- class `std::binary_negate<_Predicate>`
- class `std::unary_negate<_Predicate>`

Functions

- template<typename _Predicate>
constexpr `unary_negate<_Predicate>` `std::not1` (const _Predicate &__pred)
- template<typename _Predicate>
constexpr `binary_negate<_Predicate>` `std::not2` (const _Predicate &__pred)

2.54.1 Detailed Description

The functions `not1` and `not2` each take a predicate functor and return an instance of `unary_negate` or `binary_negate`, respectively. These classes are functors whose `operator()` performs the stored predicate function and then returns the negation of the result.

For example, given a vector of integers and a trivial predicate,

```

struct IntGreaterThanThree
: public std::unary_function<int, bool>
{
    bool operator() (int x) { return x > 3; }
};
std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
  
```

The call to `find_if` will locate the first index (*i*) of *v* for which `!(v[i] > 3)` is true.

The `not1/unary_negate` combination works on predicates taking a single argument. The `not2/binary_negate` combination works on predicates which take two arguments.

2.54.2 Function Documentation

2.54.2.1 not1()

```
template<typename _Predicate >  
constexpr unary_negate<_Predicate> std::not1 (  
    const _Predicate & __pred ) [inline]
```

One of the [negation functors](#).

Definition at line 1024 of file stl_function.h.

2.54.2.2 not2()

```
template<typename _Predicate >  
constexpr binary_negate<_Predicate> std::not2 (  
    const _Predicate & __pred ) [inline]
```

One of the [negation functors](#).

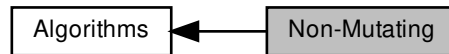
Definition at line 1052 of file stl_function.h.

2.55 Networking-ts

2.55.1 Detailed Description

2.56 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`
`constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Tp >`
`constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp & __value)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`constexpr bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2 >`
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`
`constexpr bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Tp >`
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)`

- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`
`constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2,`
`_ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`
`constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`
`constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`
`_InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`
`constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,`
`_ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count,`
`const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`
`constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count,`
`const _Tp & __val, _BinaryPredicate __binary_pred)`

2.56.1 Detailed Description

2.56.2 Function Documentation

2.56.2.1 `adjacent_find()` [1/2]

```
template<typename _ForwardIterator >
constexpr _ForwardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Find two adjacent values in a sequence that are equal.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `*i == *(i+1)`, or `__last` if no such iterator exists.

Definition at line 4031 of file `stl_algo.h`.

2.56.2.2 `adjacent_find()` [2/2]

```
template<typename _ForwardIterator , typename _BinaryPredicate >
constexpr _ForwardIterator std::adjacent_find (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _BinaryPredicate __binary_pred ) [inline]
```

Find two adjacent values in a sequence using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__binary_pred</code>	A binary predicate.

Returns

The first iterator `i` such that `i` and `i+1` are both valid iterators in `[__first,__last)` and such that `__binary_pred(i,i+1)` is true, or `__last` if no such iterator exists.

Definition at line 4057 of file `stl_algo.h`.

2.56.2.3 all_of()

```
template<typename _InputIterator , typename _Predicate >
constexpr bool std::all_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is true for all the elements of a sequence.

Parameters

<i>__first</i>	An input iterator.
<i>__last</i>	An input iterator.
<i>__pred</i>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if *__pred* is true for each element in the range [*__first*,*__last*), and false otherwise.

Definition at line 452 of file `stl_algo.h`.

References `std::find_if_not()`.

2.56.2.4 any_of()

```
template<typename _InputIterator , typename _Predicate >
constexpr bool std::any_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is true for at least one element of a sequence.

Parameters

<i>__first</i>	An input iterator.
<i>__last</i>	An input iterator.
<i>__pred</i>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if an element exists in the range [*__first*,*__last*) such that *__pred* is true, and false otherwise.

Definition at line 489 of file `stl_algo.h`.

References `std::none_of()`.

2.56.2.5 `count()`

```
template<typename _InputIterator , typename _Tp >
constexpr iterator\_traits<_InputIterator>::difference_type std::count (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __value ) [inline]
```

Count the number of copies of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__value</code>	The value to be counted.

Returns

The number of iterators `i` in the range `[__first,__last)` for which `*i == __value`

Definition at line 4083 of file `stl_algo.h`.

2.56.2.6 `count_if()`

```
template<typename _InputIterator , typename _Predicate >
constexpr iterator\_traits<_InputIterator>::difference_type std::count_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Count the elements of a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The number of iterators `i` in the range `[__first,__last)` for which `__pred(*i)` is true.

Definition at line 4107 of file `stl_algo.h`.

2.56.2.7 equal() [1/4]

```
template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
constexpr bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _BinaryPredicate __binary_pred ) [inline]
```

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1435 of file `stl_algobase.h`.

2.56.2.8 equal() [2/4]

```
template<typename _II1 , typename _II2 >
constexpr bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2 ) [inline]
```

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1404 of file `stl_algobase.h`.

2.56.2.9 `equal()` [3/4]

```
template<typename _II1 , typename _II2 >
constexpr bool std::equal (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2 ) [inline]
```

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

A boolean true or false.

This compares the elements of two ranges using `==` and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1525 of file `stl_algobase.h`.

2.56.2.10 `equal()` [4/4]

```
template<typename _IIter1 , typename _IIter2 , typename _BinaryPredicate >
constexpr bool std::equal (
    _IIter1 __first1,
    _IIter1 __last1,
    _IIter2 __first2,
    _IIter2 __last2,
    _BinaryPredicate __binary_pred ) [inline]
```

Tests a range for element-wise equality.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A boolean true or false.

This compares the elements of two ranges using the `binary_pred` parameter, and returns true or false depending on whether all of the corresponding elements of the ranges are equal.

Definition at line 1558 of file `stl_algobase.h`.

Referenced by `std::operator==()`.

2.56.2.11 find()

```
template<typename _InputIterator , typename _Tp >
constexpr _InputIterator std::find (
    _InputIterator __first,
    _InputIterator __last,
    const _Tp & __val ) [inline]
```

Find the first occurrence of a value in a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first,__last)` such that `*i == __val`, or `__last` if no such iterator exists.

Definition at line 3900 of file `stl_algo.h`.

2.56.2.12 `find_end()` [1/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr _ForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Find last matching subsequence in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.

Returns

The last iterator `i` in the range `[__first1, __last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0, __last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` and returns an iterator to the `__first` element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first1, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1-(__last2-__first2))`.

Definition at line 367 of file `stl_algo.h`.

2.56.2.13 `find_end()` [2/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
constexpr _ForwardIterator1 std::find_end (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __comp ) [inline]
```

Find last matching subsequence in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of sequence to match.
<code>__last2</code>	End of sequence to match.
<code>__comp</code>	The predicate to use.

Returns

The last iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `predicate(*(i+N), (__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)` using `comp` as a predicate and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found. The sub-sequence will be the last such subsequence contained in `[__first, __last1)`.

Because the sub-sequence must lie completely within the range `[__first1, __last1)` it must start at a position less than `__last1 - (__last2 - __first2)` where `__last2 - __first2` is the length of the sub-sequence. This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 417 of file `stl_algo.h`.

2.56.2.14 find_first_of() [1/2]

```
template<typename _InputIterator, typename _ForwardIterator >
constexpr _InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2 )
```

Find element from a set in a sequence.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `*i == *(i2)` such that `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 3957 of file `stl_algo.h`.

2.56.2.15 `find_first_of()` [2/2]

```
template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
constexpr _InputIterator std::find_first_of (
    _InputIterator __first1,
    _InputIterator __last1,
    _ForwardIterator __first2,
    _ForwardIterator __last2,
    _BinaryPredicate __comp )
```

Find element from a set in a sequence using a predicate.

Parameters

<code>__first1</code>	Start of range to search.
<code>__last1</code>	End of range to search.
<code>__first2</code>	Start of match candidates.
<code>__last2</code>	End of match candidates.
<code>__comp</code>	Predicate to use.

Returns

The first iterator `i` in the range `[__first1, __last1)` such that `comp(*i, *(i2))` is true and `i2` is an iterator in `[__first2, __last2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[__first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

Definition at line 3999 of file `stl_algo.h`.

2.56.2.16 `find_if()`

```
template<typename _InputIterator , typename _Predicate >
constexpr _InputIterator std::find_if (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is true.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is true, or `__last` if no such iterator exists.

Definition at line 3925 of file `stl_algo.h`.

Referenced by `std::none_of()`.

2.56.2.17 find_if_not()

```
template<typename _InputIterator , typename _Predicate >
constexpr _InputIterator std::find_if_not (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Find the first element in a sequence for which a predicate is false.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

The first iterator `i` in the range `[__first,__last)` such that `__pred(*i)` is false, or `__last` if no such iterator exists.

Definition at line 505 of file `stl_algo.h`.

Referenced by `std::all_of()`, and `std::is_partitioned()`.

2.56.2.18 for_each()

```
template<typename _InputIterator , typename _Function >
constexpr _Function std::for_each (
    _InputIterator __first,
    _InputIterator __last,
    _Function __f )
```

Apply a function to every element of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__f</code>	A unary function object.

Returns

`__f`

Applies the function object `__f` to each element in the range `[first,last)`. `__f` must not modify the order of the sequence. If `__f` has a return value it is ignored.

Definition at line 3838 of file `stl_algo.h`.

2.56.2.19 `is_permutation()` [1/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2 ) [inline]
```

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 2044 of file `stl_algobase.h`.

2.56.2.20 `is_permutation()` [2/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
constexpr bool std::is_permutation (
    _ForwardIterator1 __first1,
```

```
_ForwardIterator1 __last1,  
_ForwardIterator2 __first2,  
_BinaryPredicate __pred ) [inline]
```

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__pred</code>	A binary predicate.

Returns

true if there exists a permutation of the elements in the range `[__first2, __first2 + (__last1 - __first1))`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 3532 of file `stl_algo.h`.

2.56.2.21 `is_permutation()` [3/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr bool std::is_permutation (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Checks whether a permutaion of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.

Returns

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, begin)` returns true; otherwise, returns false.

Definition at line 3626 of file `stl_algo.h`.

2.56.2.22 `is_permutation()` [4/4]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
constexpr bool std::is_permutation (
```

```

    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __pred ) [inline]

```

Checks whether a permutation of the second sequence is equal to the first sequence.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of first range.
<code>__pred</code>	A binary predicate.

Returns

true if there exists a permutation of the elements in the range `[__first2, __last2)`, beginning with `ForwardIterator2` begin, such that `equal(__first1, __last1, __begin, __pred)` returns true; otherwise, returns false.

Definition at line 3655 of file `stl_algo.h`.

2.56.2.23 mismatch() [1/4]

```

template<typename _InputIterator1 , typename _InputIterator2 >
constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2 ) [inline]

```

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1768 of file `stl_algobase.h`.

2.56.2.24 mismatch() [2/4]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _BinaryPredicate __binary_pred ) [inline]
```

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1803 of file `stl_algobase.h`.

2.56.2.25 mismatch() [3/4]

```
template<typename _InputIterator1 , typename _InputIterator2 >
constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 ) [inline]
```

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using `==` and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1851 of file `stl_algobase.h`.

2.56.2.26 mismatch() [4/4]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _BinaryPredicate >
constexpr pair<_InputIterator1, _InputIterator2> std::mismatch (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _BinaryPredicate __binary_pred ) [inline]
```

Finds the places in ranges which don't match.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__binary_pred</code>	A binary predicate functor .

Returns

A pair of iterators pointing to the first mismatch.

This compares the elements of two ranges using the `binary_pred` parameter, and returns a pair of iterators. The first iterator points into the first range, the second iterator points into the second range, and the elements pointed to by the iterators are not equal.

Definition at line 1888 of file `stl_algobase.h`.

2.56.2.27 none_of()

```
template<typename _InputIterator , typename _Predicate >
constexpr bool std::none_of (
    _InputIterator __first,
    _InputIterator __last,
    _Predicate __pred ) [inline]
```

Checks that a predicate is false for all the elements of a sequence.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	An input iterator.
<code>__pred</code>	A predicate.

Returns

True if the check is true, false otherwise.

Returns true if `__pred` is false for each element in the range `[__first,__last)`, and false otherwise.

Definition at line 470 of file `stl_algo.h`.

References `std::find_if()`.

Referenced by `std::any_of()`, and `std::is_partitioned()`.

2.56.2.28 `search()` [1/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 >
constexpr _ForwardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2 ) [inline]
```

Search a sequence for a matching sub-sequence.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.

Returns

The first iterator `i` in the range `[__first1,__last1-(__last2-__first2))` such that `*(i+N) == *(__first2+N)` for each `N` in the range `[0,__last2-__first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1,__last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2,__last2)` and returns an iterator to the first element of the sub-sequence, or `__last1` if the sub-sequence is not found.

Because the sub-sequence must lie completely within the range `[__first1,__last1)` it must start at a position less than `__last1-(__last2-__first2)` where `__last2-__first2` is the length of the sub-sequence.

This means that the returned iterator `i` will be in the range `[__first1, __last1 - (__last2 - __first2))`

Definition at line 4148 of file `stl_algo.h`.

2.56.2.29 `search()` [2/2]

```
template<typename _ForwardIterator1 , typename _ForwardIterator2 , typename _BinaryPredicate >
constexpr _ForwardIterator1 std::search (
    _ForwardIterator1 __first1,
    _ForwardIterator1 __last1,
    _ForwardIterator2 __first2,
    _ForwardIterator2 __last2,
    _BinaryPredicate __predicate ) [inline]
```

Search a sequence for a matching sub-sequence using a predicate.

Parameters

<code>__first1</code>	A forward iterator.
<code>__last1</code>	A forward iterator.
<code>__first2</code>	A forward iterator.
<code>__last2</code>	A forward iterator.
<code>__predicate</code>	A binary predicate.

Returns

The first iterator `i` in the range `[__first1, __last1 - (__last2 - __first2))` such that `__predicate(*(i+N),*(__first2+N))` is true for each `N` in the range `[0, __last2 - __first2)`, or `__last1` if no such iterator exists.

Searches the range `[__first1, __last1)` for a sub-sequence that compares equal value-by-value with the sequence given by `[__first2, __last2)`, using `__predicate` to determine equality, and returns an iterator to the first element of the sub-sequence, or `__last1` if no such iterator exists.

See also

`search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)`

Definition at line 4189 of file `stl_algo.h`.

2.56.2.30 `search_n()` [1/2]

```
template<typename _ForwardIterator , typename _Integer , typename _Tp >
constexpr _ForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val ) [inline]
```

Search a sequence for a number of consecutive values.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.

Returns

The first iterator `i` in the range `[__first, __last - __count)` such that `*(i+N) == __val` for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `count` consecutive elements equal to `__val`.

Definition at line 4224 of file `stl_algo.h`.

2.56.2.31 `search_n()` [2/2]

```
template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
constexpr _ForwardIterator std::search_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Integer __count,
    const _Tp & __val,
    _BinaryPredicate __binary_pred ) [inline]
```

Search a sequence for a number of consecutive values using a predicate.

Parameters

<code>__first</code>	A forward iterator.
<code>__last</code>	A forward iterator.
<code>__count</code>	The number of consecutive values.
<code>__val</code>	The value to find.
<code>__binary_pred</code>	A binary predicate.

Returns

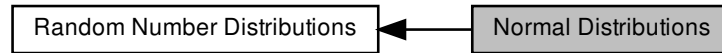
The first iterator `i` in the range `[__first, __last - __count)` such that `__binary_pred(*(i+N), __val)` is true for each `N` in the range `[0, __count)`, or `__last` if no such iterator exists.

Searches the range `[__first, __last)` for `__count` consecutive elements for which the predicate returns true.

Definition at line 4259 of file `stl_algo.h`.

2.57 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class `std::cauchy_distribution<_RealType>`
- class `std::chi_squared_distribution<_RealType>`
- class `std::fisher_f_distribution<_RealType>`
- class `std::gamma_distribution<_RealType>`
- class `std::lognormal_distribution<_RealType>`
- class `std::normal_distribution<_RealType>`
- struct `std::student_t_distribution<_RealType>::param_type`
- struct `std::fisher_f_distribution<_RealType>::param_type`
- struct `std::cauchy_distribution<_RealType>::param_type`
- struct `std::chi_squared_distribution<_RealType>::param_type`
- struct `std::gamma_distribution<_RealType>::param_type`
- struct `std::lognormal_distribution<_RealType>::param_type`
- struct `std::normal_distribution<_RealType>::param_type`
- class `std::student_t_distribution<_RealType>`

Functions

- template<typename `_RealType`>
bool `std::operator!=` (const `std::normal_distribution<_RealType>` &__d1, const `std::normal_distribution<_RealType>` &__d2)
- template<typename `_RealType`>
bool `std::operator!=` (const `std::lognormal_distribution<_RealType>` &__d1, const `std::lognormal_distribution<_RealType>` &__d2)
- template<typename `_RealType`>
bool `std::operator!=` (const `std::gamma_distribution<_RealType>` &__d1, const `std::gamma_distribution<_RealType>` &__d2)
- template<typename `_RealType`>
bool `std::operator!=` (const `std::chi_squared_distribution<_RealType>` &__d1, const `std::chi_squared_distribution<_RealType>` &__d2)
- template<typename `_RealType`>
bool `std::operator!=` (const `std::cauchy_distribution<_RealType>` &__d1, const `std::cauchy_distribution<_RealType>` &__d2)

- `template<typename _RealType >`
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`

2.57.1 Detailed Description

2.57.2 Function Documentation

2.57.2.1 `operator!=()` [1/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::normal_distribution< _RealType > & __d1,
    const std::normal_distribution< _RealType > & __d2 ) [inline]
```

Return true if two normal distributions are different.

Definition at line 2176 of file random.h.

2.57.2.2 `operator!=()` [2/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::lognormal_distribution< _RealType > & __d1,
    const std::lognormal_distribution< _RealType > & __d2 ) [inline]
```

Return true if two lognormal distributions are different.

Definition at line 2387 of file random.h.

2.57.2.3 operator!=() [3/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::gamma_distribution< _RealType > & __d1,
    const std::gamma_distribution< _RealType > & __d2 ) [inline]
```

Return true if two gamma distributions are different.

Definition at line 2618 of file random.h.

2.57.2.4 operator!=() [4/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::chi_squared_distribution< _RealType > & __d1,
    const std::chi_squared_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Chi-squared distributions are different.

Definition at line 2842 of file random.h.

2.57.2.5 operator!=() [5/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::cauchy_distribution< _RealType > & __d1,
    const std::cauchy_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Cauchy distributions have different parameters.

Definition at line 3016 of file random.h.

2.57.2.6 operator!=() [6/7]

```
template<typename _RealType >
bool std::operator!= (
    const std::fisher_f_distribution< _RealType > & __d1,
    const std::fisher_f_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Fisher f distributions are different.

Definition at line 3280 of file random.h.

2.57.2.7 operator!=() [7/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::student_t_distribution< _RealType > & __d1,
    const std::student_t_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Student t distributions are different.

Definition at line 3502 of file random.h.

2.57.2.8 operator<<()

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::cauchy_distribution< _RealType > & __x )
```

Inserts a cauchy_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A cauchy_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 2110 of file bits/random.tcc.

References std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::precision(), and std::basic_istream< _CharT, _Traits >::widen().

2.57.2.9 operator>>()

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::cauchy_distribution< _RealType > & __x )
```

Extracts a cauchy_distribution random number distribution __x from the input stream __is.

Parameters

<code>_↔ _is</code>	An input stream.
<code>_↔ _x</code>	A <code>cauchy_distribution</code> random number generator engine.

Returns

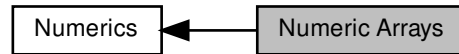
The input stream with `__x` extracted or in an error state.

Definition at line 2133 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::cauchy_distribution<_RealType>::param()`, and `std::skipws()`.

2.58 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class `std::gslice`
- class `std::gslice_array< _Tp >`
- class `std::indirect_array< _Tp >`
- class `std::mask_array< _Tp >`
- class `std::slice`
- class `std::slice_array< _Tp >`
- class `std::valarray< _Tp >`

Macros

- `#define _DEFINE_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_EXPR_AUGMENTED_ASSIGNMENT(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_VALARRAY_UNARY_OPERATOR(_Op, _Name)`

Functions

- `std::gslice::gslice ()`
- `std::gslice::gslice (size_t __o, const valarray< size_t > & __l, const valarray< size_t > & __s)`
- `std::gslice::gslice (const gslice &)`
- `std::gslice_array< _Tp >::gslice_array (const gslice_array &)`
- `std::indirect_array< _Tp >::indirect_array (const indirect_array &)`
- `std::mask_array< _Tp >::mask_array (const mask_array &)`
- `std::slice::slice ()`
- `std::slice::slice (size_t __o, size_t __d, size_t __s)`
- `std::slice_array< _Tp >::slice_array (const slice_array &)`
- `std::valarray< _Tp >::valarray ()`
- `std::valarray< _Tp >::valarray (size_t)`

- `std::valarray<_Tp>::valarray` (const `_Tp` &, `size_t`)
- `std::valarray<_Tp>::valarray` (const `valarray` &)
- `std::valarray<_Tp>::valarray` (`valarray` &&) noexcept
- `std::valarray<_Tp>::valarray` (const `slice_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `gslice_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `mask_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (const `indirect_array<_Tp>` &)
- `std::valarray<_Tp>::valarray` (`initializer_list<_Tp>`)
- `template<class _Dom>`
`std::valarray<_Tp>::valarray` (const `_Expr<_Dom, _Tp>` & `__e`)
- `template<typename _Tp>`
`std::valarray<_Tp>::valarray` (const `_Tp *` `__restrict` `__p`, `size_t` `__n`)
- `std::gslice::~gslice` ()
- `_Expr<_ValFunClos<_ValArray, _Tp>, _Tp>` `std::valarray<_Tp>::apply` (`_Tp` `func(_Tp)`) const
- `_Expr<_RefFunClos<_ValArray, _Tp>, _Tp>` `std::valarray<_Tp>::apply` (`_Tp` `func(const _Tp &)`) const
- `template<class _Tp>`
`_Tp *` `std::begin` (`valarray<_Tp>` & `__va`)
- `template<class _Tp>`
`const _Tp *` `std::begin` (const `valarray<_Tp>` & `__va`)
- `valarray<_Tp>` `std::valarray<_Tp>::cshift` (int `__n`) const
- `template<class _Tp>`
`_Tp *` `std::end` (`valarray<_Tp>` & `__va`)
- `template<class _Tp>`
`const _Tp *` `std::end` (const `valarray<_Tp>` & `__va`)
- `_Tp` `std::valarray<_Tp>::max` () const
- `_Tp` `std::valarray<_Tp>::min` () const
- `_UnaryOp<__logical_not>::Rt` `std::valarray<_Tp>::operator!` () const
- `template<typename _Tp>`
`_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type>` `std::operator!=` (const `valarray<_Tp>` & `__v`, const `typename valarray<_Tp>::value_type` & `__t`)
- `template<typename _Tp>`
`_Expr<_BinClos<__not_equal_to, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type>` `std::operator!=` (const `typename valarray<_Tp>::value_type` & `__t`, const `valarray<_Tp>` & `__v`)
- `template<typename _Tp>`
`_Expr<_BinClos<__not_equal_to, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type>` `std::operator!=` (const `valarray<_Tp>` & `__v`, const `valarray<_Tp>` & `__w`)
- `template<typename _Tp>`
`_Expr<_BinClos<__modulus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type>` `std::operator%` (const `valarray<_Tp>` & `__v`, const `valarray<_Tp>` & `__w`)
- `template<typename _Tp>`
`_Expr<_BinClos<__modulus, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type>` `std::operator%` (const `valarray<_Tp>` & `__v`, const `typename valarray<_Tp>::value_type` & `__t`)
- `template<typename _Tp>`
`_Expr<_BinClos<__modulus, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__modulus, _Tp>::result_type>` `std::operator%` (const `typename valarray<_Tp>::value_type` & `__t`, const `valarray<_Tp>` & `__v`)
- void `std::gslice_array<_Tp>::operator%=` (const `valarray<_Tp>` &) const
- void `std::mask_array<_Tp>::operator%=` (const `valarray<_Tp>` &) const
- void `std::indirect_array<_Tp>::operator%=` (const `valarray<_Tp>` &) const

- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator%= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator%= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator%= (const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator%= (const valarray<_Tp> &)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator%= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__bitwise_and, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator& (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__bitwise_and, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator& (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__bitwise_and, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__bitwise_and, _Tp>::result_type> std::operator& (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_and, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&& (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_and, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&& (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__logical_and, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__logical_and, _Tp>::result_type> std::operator&& (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `void std::gslice_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator&= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator&= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator&= (const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator&= (const valarray<_Tp> &)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator&= (const _Expr<_Dom, _Tp> &)`

- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator*= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator*= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator*= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator*= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator*= (const _Expr< _Dom, _Tp > &)`
- `_UnaryOp< __unary_plus >::Rt std::valarray< _Tp >::operator+ () const`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > std::operator+ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator+= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator+= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator+= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator+= (const valarray< _Tp > &)`

- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator+= (const _Expr<_Dom, _Tp> &)`
- `_UnaryOp<__negate>::Rt std::valarray<_Tp>::operator- () const`
- `template<typename _Tp >`
`_Expr<_BinClos<__minus, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type >`
`> std::operator- (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__minus, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type >`
`> std::operator- (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__minus, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__minus, _Tp>::result_type >`
`> std::operator- (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `void std::gslice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__divides, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type >`
`> std::operator/ (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__divides, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type >`
`> std::operator/ (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__divides, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__divides, _Tp>::result_type >`
`> std::operator/ (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `void std::gslice_array<_Tp>::operator/= (const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator/= (const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator/= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator/= (const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator/= (const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator/= (const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator/= (const valarray<_Tp> &)`

- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator/=(const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type >`
`std::operator<(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type >`
`std::operator<(const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__less, _Tp>::result_type >`
`std::operator<(const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__shift_left, _Tp>::result_type >`
`std::operator<<(const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__shift_left, _Tp>::result_type >`
`std::operator<<(const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp >`
`_Expr<_BinClos<__shift_left, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__shift_left, _Tp>::result_type >`
`std::operator<<(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `void std::gslice_array<_Tp>::operator<=<(const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator<=<(const valarray<_Tp> &) const`
- `void std::indirect_array<_Tp>::operator<=<(const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::gslice_array<_Tp>::operator<=<(const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::indirect_array<_Tp>::operator<=<(const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom >`
`void std::mask_array<_Tp>::operator<=<(const _Expr<_Dom, _Tp> &) const`
- `void std::slice_array<_Tp>::operator<=<(const valarray<_Tp> &) const`
- `template<class _Dom >`
`void std::slice_array<_Tp>::operator<=<(const _Expr<_Dom, _Tp> &) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator<=<(const _Tp &)`
- `valarray<_Tp> & std::valarray<_Tp>::operator<=<(const valarray<_Tp> &)`
- `template<class _Dom >`
`valarray<_Tp> & std::valarray<_Tp>::operator<=<(const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__less_equal, _Tp>::result_type >`
`std::operator<=<(const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__less_equal, _Tp>::result_type >`
`std::operator<=<(const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp >`
`_Expr<_BinClos<__less_equal, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__less_equal, _Tp>::result_type >`
`std::operator<=<(const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `gslice_array & std::gslice_array<_Tp>::operator=(const gslice_array &)`
- `indirect_array & std::indirect_array<_Tp>::operator=(const indirect_array &)`
- `mask_array & std::mask_array<_Tp>::operator=(const mask_array &)`
- `void std::gslice_array<_Tp>::operator=(const valarray<_Tp> &) const`
- `void std::mask_array<_Tp>::operator=(const valarray<_Tp> &) const`

- `void std::indirect_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `gslice & std::gslice::operator= (const gslice &)`
- `void std::gslice_array<_Tp>::operator= (const _Tp &) const`
- `void std::mask_array<_Tp>::operator= (const _Tp &) const`
- `void std::indirect_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom>`
`void std::gslice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Dom>`
`void std::indirect_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `slice_array & std::slice_array<_Tp>::operator= (const slice_array &)`
- `void std::slice_array<_Tp>::operator= (const valarray<_Tp> &) const`
- `void std::slice_array<_Tp>::operator= (const _Tp &) const`
- `template<class _Dom>`
`void std::slice_array<_Tp>::operator= (const _Expr<_Dom, _Tp> &) const`
- `template<class _Ex>`
`void std::mask_array<_Tp>::operator= (const _Expr<_Ex, _Tp> &__e) const`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const valarray<_Tp> &__v)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (valarray<_Tp> &&__v) noexcept`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const _Tp &__t)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & std::valarray<_Tp>::operator= (const indirect_array<_Tp> &__ia)`
- `valarray & std::valarray<_Tp>::operator= (initializer_list<_Tp> &__l)`
- `template<class _Dom>`
`valarray<_Tp> & std::valarray<_Tp>::operator= (const _Expr<_Dom, _Tp> &)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`
`_Expr<_BinClos<__equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__equal_to, _Tp>::result_type> std::operator== (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _ValArray, _ValArray, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp> &__v, const valarray<_Tp> &__w)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__greater, _Tp>::result_type> std::operator> (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`
- `template<typename _Tp>`
`_Expr<_BinClos<__greater_equal, _Constant, _ValArray, _Tp, _Tp>, typename __fun<__greater_equal, _Tp>::result_type> std::operator>= (const typename valarray<_Tp>::value_type &__t, const valarray<_Tp> &__v)`

- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > std::operator>= (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > std::operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator>>= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator>>= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator>>= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator>>= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator>>= (const _Expr< _Dom, _Tp > &)`
- `_Tp & std::valarray< _Tp >::operator[] (size_t __i)`
- `const _Tp & std::valarray< _Tp >::operator[] (size_t) const`
- `_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (slice __s) const`
- `slice_array< _Tp > std::valarray< _Tp >::operator[] (slice __s)`
- `_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const gslice &__s) const`
- `gslice_array< _Tp > std::valarray< _Tp >::operator[] (const gslice &__s)`
- `valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m) const`
- `mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m)`
- `_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i) const`
- `indirect_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`

- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `void std::gslice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator^= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator^= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator^= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator^= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > std::operator| (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `void std::gslice_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::mask_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `void std::indirect_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::gslice_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::indirect_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `template<class _Dom >`
`void std::mask_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `void std::slice_array< _Tp >::operator|= (const valarray< _Tp > &) const`
- `template<class _Dom >`
`void std::slice_array< _Tp >::operator|= (const _Expr< _Dom, _Tp > &) const`
- `valarray< _Tp > & std::valarray< _Tp >::operator|= (const _Tp &)`
- `valarray< _Tp > & std::valarray< _Tp >::operator|= (const valarray< _Tp > &)`
- `template<class _Dom >`
`valarray< _Tp > & std::valarray< _Tp >::operator|= (const _Expr< _Dom, _Tp > &)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >::result_type > std::operator|| (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`

- `template<typename _Tp >
_Expr< __BinClos< __logical_or, __Constant, __ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp > <←
::result_type > std::operator|| (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
__v)`
- `_UnaryOp< __bitwise_not >::Rt std::valarray< _Tp >::operator~ () const`
- `void std::valarray< _Tp >::resize (size_t __size, _Tp __c=_Tp())`
- `valarray< _Tp > std::valarray< _Tp >::shift (int __n) const`
- `size_t std::slice::size () const`
- `valarray< size_t > std::gslice::size () const`
- `size_t std::valarray< _Tp >::size () const`
- `size_t std::slice::start () const`
- `size_t std::gslice::start () const`
- `size_t std::slice::stride () const`
- `valarray< size_t > std::gslice::stride () const`
- `_Tp std::valarray< _Tp >::sum () const`
- `void std::valarray< _Tp >::swap (valarray< _Tp > &__v) noexcept`

2.58.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

2.58.2 Function Documentation

2.58.2.1 `gslice()` [1/3]

```
std::gslice::gslice ( ) [inline]
```

Construct an empty slice.

Definition at line 149 of file `gslice.h`.

2.58.2.2 `gslice()` [2/3]

```
std::gslice::gslice (
    size_t __o,
    const valarray< size_t > & __l,
    const valarray< size_t > & __s ) [inline]
```

Construct a slice.

Constructs a slice with as many dimensions as the length of the `l` and `s` arrays.

Parameters

<code>_↔ _o</code>	Offset in array of first element.
<code>_↔ _l</code>	Array of dimension lengths.
<code>_↔ _s</code>	Array of dimension strides between array elements.

Definition at line 153 of file `gslice.h`.

2.58.2.3 `gslice()` [3/3]

```
std::gslice::gslice (
    const gslice & __g ) [inline]
```

Copy constructor.

Definition at line 158 of file `gslice.h`.

2.58.2.4 `gslice_array()`

```
template<typename _Tp >
std::gslice_array< _Tp >::gslice_array (
    const gslice_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 148 of file `gslice_array.h`.

2.58.2.5 `indirect_array()`

```
template<typename _Tp >
std::indirect_array< _Tp >::indirect_array (
    const indirect_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 143 of file `indirect_array.h`.

2.58.2.6 mask_array()

```
template<typename _Tp >
std::mask_array< _Tp >::mask_array (
    const mask_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 144 of file mask_array.h.

2.58.2.7 slice() [1/2]

```
std::slice::slice ( ) [inline]
```

Construct an empty slice.

Definition at line 95 of file slice_array.h.

2.58.2.8 slice() [2/2]

```
std::slice::slice (
    size_t __o,
    size_t __d,
    size_t __s ) [inline]
```

Construct a slice.

Parameters

\leftrightarrow __o	Offset in array of first element.
\leftrightarrow __d	Number of elements in slice.
\leftrightarrow __s	Stride between array elements.

Definition at line 99 of file slice_array.h.

2.58.2.9 slice_array()

```
template<typename _Tp >
std::slice_array< _Tp >::slice_array (
    const slice_array< _Tp > & __a ) [inline]
```

Copy constructor. Both slices refer to the same underlying array.

Definition at line 217 of file slice_array.h.

2.58.2.10 valarray() [1/10]

```
template<typename _Tp >
std::valarray< _Tp >::valarray ( ) [inline]
```

Construct an empty array.

Definition at line 621 of file valarray.

2.58.2.11 valarray() [2/10]

```
template<typename _Tp >
std::valarray< _Tp >::valarray (
    size_t __n ) [inline], [explicit]
```

Construct an array with n elements.

Definition at line 625 of file valarray.

2.58.2.12 valarray() [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const _Tp & __t,
    size_t __n ) [inline]
```

Construct an array with n elements initialized to t .

Definition at line 631 of file valarray.

2.58.2.13 valarray() [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const valarray< _Tp > & __v ) [inline]
```

Copy constructor.

Definition at line 646 of file valarray.

2.58.2.14 `valarray()` [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    valarray< _Tp > && __v ) [inline], [noexcept]
```

Move constructor.

Definition at line 654 of file `valarray`.

2.58.2.15 `valarray()` [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const slice_array< _Tp > & __sa ) [inline]
```

Construct an array with the same size and values in *sa*.

Definition at line 664 of file `valarray`.

2.58.2.16 `valarray()` [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const gslice_array< _Tp > & __ga ) [inline]
```

Construct an array with the same size and values in *ga*.

Definition at line 673 of file `valarray`.

2.58.2.17 `valarray()` [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const mask_array< _Tp > & __ma ) [inline]
```

Construct an array with the same size and values in *ma*.

Definition at line 684 of file `valarray`.

2.58.2.18 `valarray()` [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    const indirect_array< _Tp > & __ia ) [inline]
```

Construct an array with the same size and values in *ia*.

Definition at line 693 of file `valarray`.

2.58.2.19 `valarray()` [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
    initializer_list< _Tp > __l ) [inline]
```

Construct an array with an `initializer_list` of values.

Definition at line 703 of file `valarray`.

2.58.2.20 `~gslice()`

```
std::gslice::~gslice ( ) [inline]
```

Destructor.

Definition at line 163 of file `gslice.h`.

2.58.2.21 `apply()` [1/2]

```
template<class _Tp>
_Expr< _ValFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp func_Tp ) const [inline]
```

Apply a function to the array.

Returns a new `valarray` with elements assigned to the result of applying `func` to the corresponding element of this array. The new array has the same size as this one.

Parameters

<i>func</i>	Function of <code>Tp</code> returning <code>Tp</code> to apply.
-------------	---

Returns

New valarray with transformed elements.

Definition at line 1065 of file valarray.

2.58.2.22 apply() [2/2]

```
template<class _Tp>
_Expr< _RefFuncClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
    _Tp funcconst _Tp & ) const [inline]
```

Apply a function to the array.

Returns a new valarray with elements assigned to the result of applying func to the corresponding element of this array. The new array has the same size as this one.

Parameters

<i>func</i>	Function of const Tp& returning Tp to apply.
-------------	--

Returns

New valarray with transformed elements.

Definition at line 1073 of file valarray.

2.58.2.23 begin() [1/2]

```
template<class _Tp >
_Tp * std::begin (
    valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to the first element of the valarray.

Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1214 of file valarray.

Referenced by std::cbegin(), std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert(), std::list< __inp, __rebind_inp >::merge(), std::list< __inp, __rebind_inp >::remove(), std::list< __inp, __rebind_inp >::remove_if(), and std::list< __inp, __rebind_inp >::unique().

2.58.2.24 begin() [2/2]

```
template<class _Tp >
const _Tp * std::begin (
    const valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to the first element of the const valarray.

Parameters

<code>__va</code>	valarray.
-------------------	-----------

Definition at line 1224 of file valarray.

2.58.2.25 cshift()

```
template<class _Tp >
valarray< _Tp > std::valarray< _Tp >::cshift (
    int __n ) const [inline]
```

Return a rotated array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index *i*, the new position is $(i - n) \% \text{size}()$. The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.

Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

<code>__n</code>	Number of element positions to rotate.
------------------	--

Returns

New valarray with elements in shifted positions.

Definition at line 991 of file valarray.

2.58.2.26 end() [1/2]

```
template<class _Tp >
_Tp * std::end (
    valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to one past the last element of the valarray.

Parameters

<code>__va</code>	<code>valarray.</code>
-------------------	------------------------

Definition at line 1234 of file `valarray`.

Referenced by `std::cend()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`, `std::list< __inp, __rebind_inp >::merge()`, `std::basic_filebuf< char_type, traits_type >::open()`, `std::list< __inp, __rebind_inp >::remove()`, `std::list< __inp, __rebind_inp >::remove_if()`, `std::list< __inp, __rebind_inp >::resize()`, and `std::list< __inp, __rebind_inp >::unique()`.

2.58.2.27 `end()` [2/2]

```
template<class _Tp >
const _Tp * std::end (
    const valarray< _Tp > & __va ) [inline]
```

Return an iterator pointing to one past the last element of the `const valarray`.

Parameters

<code>__va</code>	<code>valarray.</code>
-------------------	------------------------

Definition at line 1244 of file `valarray`.

2.58.2.28 `max()`

```
template<typename _Tp >
_Tp std::valarray< _Tp >::max ( ) const [inline]
```

Return the maximum element using `operator<()`.

Definition at line 1057 of file `valarray`.

2.58.2.29 `min()`

```
template<typename _Tp >
_Tp std::valarray< _Tp >::min ( ) const [inline]
```

Return the minimum element using `operator<()`.

Definition at line 1049 of file `valarray`.

2.58.2.30 `operator!()`

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator! ( )
const [inline]
```

Return a new valarray by applying unary ! to each element.

Definition at line 1092 of file valarray.

2.58.2.31 `operator%=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 207 of file gslice_array.h.

2.58.2.32 `operator%=()` [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 197 of file mask_array.h.

2.58.2.33 `operator%=()` [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 196 of file indirect_array.h.

2.58.2.34 operator%=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator%= (
    const valarray< _Tp > & __v ) const [inline]
```

Modulo slice elements by corresponding elements of *v*.

Definition at line 268 of file slice_array.h.

2.58.2.35 operator%=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e* % *t*.

Definition at line 1119 of file valarray.

2.58.2.36 operator%=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
    const valarray< _Tp > & __v ) [inline]
```

Modulo elements of array by corresponding elements of *v*.

Definition at line 1119 of file valarray.

2.58.2.37 operator&=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 211 of file gslice_array.h.

2.58.2.38 operator&=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 201 of file mask_array.h.

2.58.2.39 operator&=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 200 of file indirect_array.h.

2.58.2.40 operator&=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator&= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical and slice elements with corresponding elements of *v*.

Definition at line 272 of file slice_array.h.

2.58.2.41 operator&=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e* & *t*.

Definition at line 1121 of file valarray.

2.58.2.42 operator&=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
    const valarray< _Tp > & __v ) [inline]
```

Logical and corresponding elements of *v* with elements of array.

Definition at line 1121 of file valarray.

2.58.2.43 operator*=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 205 of file gslice_array.h.

2.58.2.44 operator*=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 195 of file mask_array.h.

2.58.2.45 operator*=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 194 of file indirect_array.h.

2.58.2.46 operator*=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator*= (
    const valarray< _Tp > & __v ) const [inline]
```

Multiply slice elements by corresponding elements of *v*.

Definition at line 266 of file slice_array.h.

2.58.2.47 operator*=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const _Tp & __t ) [inline]
```

Multiply each element of array by *t*.

Definition at line 1117 of file valarray.

2.58.2.48 operator*=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
    const valarray< _Tp > & __v ) [inline]
```

Multiply elements of array by corresponding elements of *v*.

Definition at line 1117 of file valarray.

2.58.2.49 operator+()

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ ( )
const [inline]
```

Return a new valarray by applying unary + to each element.

Definition at line 1089 of file valarray.

2.58.2.50 operator+=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 208 of file gslice_array.h.

2.58.2.51 operator+=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 198 of file mask_array.h.

2.58.2.52 operator+=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 197 of file indirect_array.h.

2.58.2.53 operator+=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator+= (
    const valarray< _Tp > & __v ) const [inline]
```

Add corresponding elements of *v* to slice elements.

Definition at line 269 of file slice_array.h.

2.58.2.54 operator+=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const _Tp & __t ) [inline]
```

Add *t* to each element of array.

Definition at line 1115 of file valarray.

2.58.2.55 operator+=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
    const valarray< _Tp > & __v ) [inline]
```

Add corresponding elements of *v* to elements of array.

Definition at line 1115 of file valarray.

2.58.2.56 operator-()

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- ( ) const
[inline]
```

Return a new valarray by applying unary - to each element.

Definition at line 1090 of file valarray.

2.58.2.57 operator-=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 209 of file gslice_array.h.

2.58.2.58 operator-=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 199 of file mask_array.h.

2.58.2.59 operator-=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 198 of file indirect_array.h.

2.58.2.60 operator-=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator-= (
    const valarray< _Tp > & __v ) const [inline]
```

Subtract corresponding elements of *v* from slice elements.

Definition at line 270 of file slice_array.h.

2.58.2.61 operator-=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const _Tp & __t ) [inline]
```

Subtract *t* to each element of array.

Definition at line 1116 of file valarray.

2.58.2.62 `operator-=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
    const valarray< _Tp > & __v ) [inline]
```

Subtract corresponding elements of *v* from elements of array.

Definition at line 1116 of file valarray.

2.58.2.63 `operator/=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 206 of file gslice_array.h.

2.58.2.64 `operator/=()` [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 196 of file mask_array.h.

2.58.2.65 `operator/=()` [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 195 of file indirect_array.h.

2.58.2.66 operator/=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator/= (
    const valarray< _Tp > & __v ) const [inline]
```

Divide slice elements by corresponding elements of *v*.

Definition at line 267 of file slice_array.h.

2.58.2.67 operator/=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const _Tp & __t ) [inline]
```

Divide each element of array by *t*.

Definition at line 1118 of file valarray.

2.58.2.68 operator/=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
    const valarray< _Tp > & __v ) [inline]
```

Divide elements of array by corresponding elements of *v*.

Definition at line 1118 of file valarray.

2.58.2.69 operator<<=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 213 of file gslice_array.h.

2.58.2.70 `operator<<=()` [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 203 of file `mask_array.h`.

2.58.2.71 `operator<<=()` [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 202 of file `indirect_array.h`.

2.58.2.72 `operator<<=()` [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) const [inline]
```

Left shift slice elements by corresponding elements of *v*.

Definition at line 274 of file `slice_array.h`.

2.58.2.73 `operator<<=()` [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const _Tp & __t ) [inline]
```

Left shift each element *e* of array by *t* bits.

Definition at line 1123 of file `valarray`.

2.58.2.74 operator<<=() [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
    const valarray< _Tp > & __v ) [inline]
```

Left shift elements of array by corresponding elements of *v*.

Definition at line 1123 of file valarray.

2.58.2.75 operator=() [1/20]

```
template<typename _Tp >
gslice_array< _Tp > & std::gslice_array< _Tp >::operator= (
    const gslice_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 153 of file gslice_array.h.

2.58.2.76 operator=() [2/20]

```
template<typename _Tp >
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
    const indirect_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file indirect_array.h.

2.58.2.77 operator=() [3/20]

```
template<typename _Tp >
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
    const mask_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.

Definition at line 154 of file mask_array.h.

2.58.2.78 operator=() [4/20]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 171 of file gslice_array.h.

References `std::valarray< _Tp >::size()`.

2.58.2.79 operator=() [5/20]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 168 of file indirect_array.h.

2.58.2.80 operator=() [6/20]

```
gslice & std::gslice::operator= (
    const gslice & __g ) [inline]
```

Assignment operator.

Definition at line 170 of file gslice.h.

2.58.2.81 operator=() [7/20]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 163 of file gslice_array.h.

2.58.2.82 operator=() [8/20]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 163 of file mask_array.h.

2.58.2.83 operator=() [9/20]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 163 of file indirect_array.h.

2.58.2.84 operator=() [10/20]

```
template<typename _Tp >
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
    const slice_array< _Tp > & __a ) [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.

Definition at line 225 of file slice_array.h.

2.58.2.85 operator=() [11/20]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator= (
    const valarray< _Tp > & __v ) const [inline]
```

Assign slice elements to corresponding elements of *v*.

Definition at line 239 of file slice_array.h.

2.58.2.86 operator=() [12/20]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator= (
    const _Tp & __t ) const [inline]
```

Assign all slice elements to *t*.

Definition at line 234 of file slice_array.h.

2.58.2.87 operator=() [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const valarray< _Tp > & __v ) [inline]
```

Assign elements to an array.

Assign elements of array to values in *v*.

Parameters

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

Definition at line 724 of file valarray.

2.58.2.88 operator=() [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    valarray< _Tp > && __v ) [inline], [noexcept]
```

Move assign elements to an array.

Move assign elements of array to values in *v*.

Parameters

<code>__v</code>	Valarray to get values from.
------------------	------------------------------

Definition at line 748 of file valarray.

2.58.2.89 operator=() [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const _Tp & __t ) [inline]
```

Assign elements to a value.

Assign all elements of array to *t*.

Parameters

↔	Value for elements.
↔	
↔	
↔	
<i>t</i>	

Definition at line 788 of file valarray.

2.58.2.90 operator=() [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const slice_array< _Tp > & __sa ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

__sa	Array slice to get values from.
------	---------------------------------

Definition at line 796 of file valarray.

2.58.2.91 operator=() [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const gslice_array< _Tp > & __ga ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

<code>__ga</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 806 of file `valarray`.

2.58.2.92 `operator=()` [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const mask_array< _Tp > & __ma ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

<code>__ma</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 816 of file `valarray`.

2.58.2.93 `operator=()` [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    const indirect_array< _Tp > & __ia ) [inline]
```

Assign elements to an array subset.

Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

<code>__ia</code>	Array slice to get values from.
-------------------	---------------------------------

Definition at line 826 of file `valarray`.

2.58.2.94 `operator=()` [20/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
    initializer_list< _Tp > __l ) [inline]
```

Assign elements to an `initializer_list`.

Assign elements of array to values in `__l`. Results are undefined if `__l` does not have the same size as this array.

Parameters

<code>↔</code>	<code>initializer_list</code> to get values from.
<code>__↔</code>	
<code>↔</code>	
<code>__↔</code>	
<code>l</code>	

Definition at line 764 of file `valarray`.

2.58.2.95 `operator>>=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) const [inline]
```

Right shift slice elements by corresponding elements of `v`.

Definition at line 214 of file `gslice_array.h`.

2.58.2.96 `operator>>=()` [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) const [inline]
```

Right shift slice elements by corresponding elements of `v`.

Definition at line 204 of file `mask_array.h`.

2.58.2.97 `operator>>=()` [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) const [inline]
```

Right shift slice elements by corresponding elements of `v`.

Definition at line 203 of file `indirect_array.h`.

2.58.2.98 `operator>>=()` [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) const [inline]
```

Right shift slice elements by corresponding elements of *v*.

Definition at line 275 of file `slice_array.h`.

2.58.2.99 `operator>>=()` [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const _Tp & __t ) [inline]
```

Right shift each element *e* of array by *t* bits.

Definition at line 1124 of file `valarray`.

2.58.2.100 `operator>>=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
    const valarray< _Tp > & __v ) [inline]
```

Right shift elements of array by corresponding elements of *v*.

Definition at line 1124 of file `valarray`.

2.58.2.101 `operator[]()` [1/9]

```
template<typename _Tp >
_Tp & std::valarray< _Tp >::operator[] (
    size_t __i ) [inline]
```

Return a reference to the *i*'th array element.

Parameters

\leftarrow	Index of element to return.
$_ \leftarrow$	
\leftarrow	
$_ \leftarrow$	
<i>i</i>	

Returns

Reference to the i'th element.

Definition at line 592 of file valarray.

2.58.2.102 operator[]() [2/9]

```
template<typename _Tp >
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    slice __s ) const [inline]
```


Return an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

 <code>__s</code>	The source slice.
--	-------------------

Returns

New valarray containing elements in `__s`.

Definition at line 858 of file valarray.

2.58.2.103 operator[]() [3/9]

```
template<typename _Tp >
slice_array< _Tp > std::valarray< _Tp >::operator[] (
    slice __s ) [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

<code>_↔</code>	The source slice.
<code>__s</code>	

Returns

New valarray containing elements in `__s`.

Definition at line 866 of file valarray.

2.58.2.104 `operator[]()` [4/9]

```
template<typename _Tp >
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s ) const [inline]
```

Return an array subset.

Returns a `slice_array` referencing the elements of the array indicated by the slice argument.

See also

`gslice`.

Parameters

<code>_↔</code>	The source slice.
<code>__s</code>	

Returns

`Slice_array` referencing elements indicated by `__s`.

Definition at line 871 of file valarray.

2.58.2.105 `operator[]()` [5/9]

```
template<typename _Tp >
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
    const gslice & __s ) [inline]
```


Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the `gslice` argument. The new valarray has the same size as the input `gslice`.

See also

gslice.

Parameters

 <u>_s</u>	The source gslice.
--	--------------------

Returns

New valarray containing elements in `__s`.

Definition at line 880 of file valarray.


2.58.2.106 operator[]() [6/9]

```
template<typename _Tp >
valarray< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m ) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

 <u>_m</u>	The valarray bitmask.
--	-----------------------

Returns

New valarray containing elements indicated by `__m`.

Definition at line 888 of file valarray.


2.58.2.107 operator[]() [7/9]

```
template<typename _Tp >
mask_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< bool > & __m ) [inline]
```

Return a reference to an array subset.

Returns a new mask_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

 <code>__m</code>	The valarray bitmask.
---	-----------------------

Returns

New valarray containing elements indicated by `__m`.

Definition at line 900 of file valarray.





2.58.2.108 `operator[]()` [8/9]

```
template<typename _Tp >
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i ) const [inline]
```

Return an array subset.

Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

	The valarray element index list.
<code>__</code> 	
	
<code>__</code>  <code>i</code>	

Returns

New valarray containing elements in `__s`.

Definition at line 911 of file valarray.






2.58.2.109 `operator[]()` [9/9]

```
template<typename _Tp >
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
    const valarray< size_t > & __i ) [inline]
```

Return a reference to an array subset.

Returns an indirect_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect_array refers to these elements.

Parameters

	The valarray element index list.
	
	
	
	

Returns

Indirect_array referencing elements in `__i`.

Definition at line 919 of file valarray.

2.58.2.110 operator^=() [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of `v`.

Definition at line 210 of file gslice_array.h.

2.58.2.111 operator^=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of `v`.

Definition at line 200 of file mask_array.h.

2.58.2.112 operator^=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of `v`.

Definition at line 199 of file indirect_array.h.

2.58.2.113 `operator^=()` [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator^= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical xor slice elements with corresponding elements of *v*.

Definition at line 271 of file slice_array.h.

2.58.2.114 `operator^=()` [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to $e \wedge t$.

Definition at line 1120 of file valarray.

2.58.2.115 `operator^=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
    const valarray< _Tp > & __v ) [inline]
```

Logical xor corresponding elements of *v* with elements of array.

Definition at line 1120 of file valarray.

2.58.2.116 `operator" |=()` [1/6]

```
template<typename _Tp >
void std::gslice_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 212 of file gslice_array.h.

2.58.2.117 operator" |=() [2/6]

```
template<typename _Tp >
void std::mask_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 202 of file mask_array.h.

2.58.2.118 operator" |=() [3/6]

```
template<typename _Tp >
void std::indirect_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 201 of file indirect_array.h.

2.58.2.119 operator" |=() [4/6]

```
template<typename _Tp >
void std::slice_array< _Tp >::operator|= (
    const valarray< _Tp > & __v ) const [inline]
```

Logical or slice elements with corresponding elements of *v*.

Definition at line 273 of file slice_array.h.

2.58.2.120 operator" |=() [5/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const _Tp & __t ) [inline]
```

Set each element *e* of array to *e* | *t*.

Definition at line 1122 of file valarray.

2.58.2.121 `operator" |=()` [6/6]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
    const valarray< _Tp > & __v ) [inline]
```

Logical or corresponding elements of `v` with elements of array.

Definition at line 1122 of file `valarray`.

2.58.2.122 `operator~()`

```
template<typename _Tp >
valarray< _Tp >::template _UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ ( )
const [inline]
```

Return a new valarray by applying unary `~` to each element.

Definition at line 1091 of file `valarray`.

2.58.2.123 `resize()`

```
template<class _Tp>
void std::valarray< _Tp >::resize (
    size_t __size,
    _Tp __c = _Tp() ) [inline]
```

Resize array.

Resize this array to `size` and set all elements to `c`. All references and iterators are invalidated.

Parameters

<code>__size</code>	New array size.
<code>__c</code>	New value for all elements.

Definition at line 1032 of file `valarray`.

2.58.2.124 `shift()`

```
template<class _Tp >
valarray< _Tp > std::valarray< _Tp >::shift (
    int __n ) const [inline]
```

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i , the new position is $i - n$. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements $[0, n)$. Negative arguments discard elements from the top of the array.

Parameters

$_n$	Number of element positions to shift.
-------	---------------------------------------

Returns

New valarray with elements in shifted positions.

Definition at line 950 of file valarray.

2.58.2.125 size() [1/3]

```
size_t std::slice::size ( ) const [inline]
```

Return size of slice.

Definition at line 107 of file slice_array.h.

2.58.2.126 size() [2/3]

```
valarray< size_t > std::gslice::size ( ) const [inline]
```

Return array of sizes of slice dimensions.

Definition at line 139 of file gslice.h.

2.58.2.127 size() [3/3]

```
template<class _Tp >  
size_t std::valarray< _Tp >::size ( ) const [inline]
```

Return the number of elements in array.

Definition at line 937 of file valarray.

Referenced by std::gslice_array< _Tp >::operator=().

2.58.2.128 start() [1/2]

```
size_t std::slice::start ( ) const [inline]
```

Return array offset of first slice element.

Definition at line 103 of file slice_array.h.

2.58.2.129 start() [2/2]

```
size_t std::gslice::start ( ) const [inline]
```

Return array offset of first slice element.

Definition at line 135 of file gslice.h.

2.58.2.130 stride() [1/2]

```
size_t std::slice::stride ( ) const [inline]
```

Return array stride of slice.

Definition at line 111 of file slice_array.h.

2.58.2.131 stride() [2/2]

```
valarray< size_t > std::gslice::stride ( ) const [inline]
```

Return array of array strides for each dimension.

Definition at line 143 of file gslice.h.

2.58.2.132 sum()

```
template<class _Tp >  
_Tp std::valarray< _Tp >::sum ( ) const [inline]
```

Return the sum of all elements in the array.

Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

Definition at line 942 of file valarray.

2.58.2.133 swap()

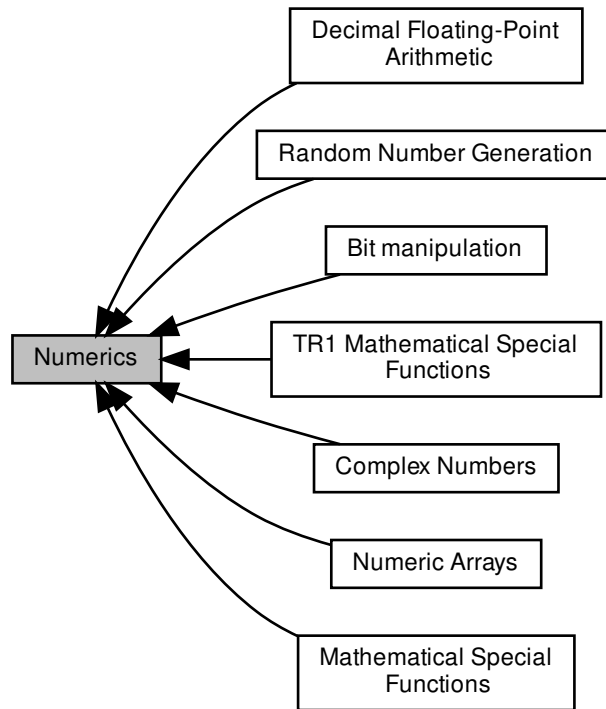
```
template<class _Tp>  
void std::valarray< _Tp >::swap (   
    valarray< _Tp > & __v ) [inline], [noexcept]
```

Swap.

Definition at line 928 of file valarray.

2.59 Numerics

Collaboration diagram for Numerics:



Modules

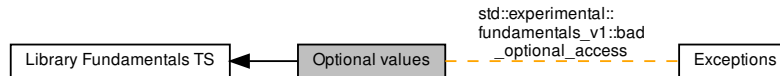
- [Bit manipulation](#)
- [Complex Numbers](#)
- [Decimal Floating-Point Arithmetic](#)
- [Mathematical Special Functions](#)
- [Numeric Arrays](#)
- [Random Number Generation](#)
- [TR1 Mathematical Special Functions](#)

2.59.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and mathematical special functions.

2.60 Optional values

Collaboration diagram for Optional values:



Classes

- class `std::experimental::fundamentals_v1::bad_optional_access`
- struct `std::experimental::fundamentals_v1::in_place_t`
- struct `std::experimental::fundamentals_v1::nullopt_t`
- class `std::experimental::fundamentals_v1::optional< _Tp >`

Macros

- `#define __cpp_lib_experimental_optional`

Variables

- constexpr `in_place_t` `std::experimental::fundamentals_v1::in_place`
- constexpr `nullopt_t` `std::experimental::fundamentals_v1::nullopt`

2.60.1 Detailed Description

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

2.60.2 Variable Documentation

2.60.2.1 in_place

```
constexpr in_place_t std::experimental::fundamentals_v1::in_place
```

Tag for in-place construction.

Definition at line 77 of file experimental/optional.

2.60.2.2 nullopt

```
constexpr nullopt_t std::experimental::fundamentals_v1::nullopt
```

Tag to disengage optional objects.

Definition at line 96 of file experimental/optional.

2.61 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- class `std::enable_shared_from_this<_Tp>`
- struct `std::hash<shared_ptr<_Tp>>`
- struct `std::hash<unique_ptr<_Tp,_Dp>>`
- struct `std::owner_less<_Tp>`
- struct `std::owner_less<shared_ptr<_Tp>>`
- struct `std::owner_less<void>`
- struct `std::owner_less<weak_ptr<_Tp>>`
- struct `std::pointer_traits<_Ptr>`
- struct `std::pointer_traits<_Tp*>`
- class `std::shared_ptr<_Tp>`
- class `std::unique_ptr<_Tp,_Dp>`
- class `std::unique_ptr<_Tp[],_Dp>`
- class `std::weak_ptr<_Tp>`

Functions

- template<typename _Del, typename _Tp, _Lock_policy _Lp>
_Del * **std::get_deleter** (const __shared_ptr<_Tp,_Lp> &__p) noexcept
- template<typename _Del, typename _Tp>
_Del * **get_deleter** (const shared_ptr<_Tp> &__p) noexcept
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
std::basic_ostream<_Ch,_Tr> & **operator<<** (std::basic_ostream<_Ch,_Tr> &__os, const __shared_ptr<_Tp,_Lp> &__p)
- template<typename _Tp>
void **swap** (weak_ptr<_Tp> &__a, weak_ptr<_Tp> &__b) noexcept
- template<typename _Tp, typename _Up>
bool **operator==** (const shared_ptr<_Tp> &__a, const shared_ptr<_Up> &__b) noexcept
- template<typename _Tp>
bool **operator==** (const shared_ptr<_Tp> &__a, nullptr_t) noexcept

- `template<typename _Tp >`
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`
`bool operator!= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`
`bool operator< (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`
`bool operator<= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`
`bool operator> (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`
`bool operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`
`bool operator>= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`
`bool operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`
`bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp >`
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Up >`
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up >`
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up >`
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`

- `template<typename _Tp, typename _Alloc, typename... _Args>`
`shared_ptr< _Tp > allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`
`shared_ptr< _Tp > make_shared (_Args &&... __args)`

- `template<typename _Tp, _Lock_policy _Lp>`
`bool atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *__p)`

- `template<typename _Tp >`
`shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)`

- `template<typename _Tp >`
`void atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp >`
`shared_ptr< _Tp > atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)`
- `template<typename _Tp >`
`bool atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)`
- `template<typename _Tp, typename _Dp >`
`enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`
`bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`
`bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`

- `template<typename _Tp, typename... _Args>`
`_MakeUniq< _Tp >::__single_object make_unique (_Args &&... __args)`
- `template<typename _Tp >`
`_MakeUniq< _Tp >::__array make_unique (size_t __num)`
- `template<typename _Tp, typename... _Args>`
`_MakeUniq< _Tp >::__invalid_type make_unique (_Args &&...)=delete`
- `#define __cpp_lib_make_unique`

2.61.1 Detailed Description

Smart pointers, etc.

2.61.2 Macro Definition Documentation

2.61.2.1 `__cpp_lib_make_unique`

```
#define __cpp_lib_make_unique
```

Definition at line 940 of file `unique_ptr.h`.

2.61.3 Function Documentation

2.61.3.1 `allocate_shared()`

```
template<typename _Tp, typename _Alloc, typename... _Args>
shared\_ptr< _Tp > allocate_shared (
    const _Alloc & __a,
    _Args &&... __args ) [related]
```

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__a</code>	An allocator.
<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<i>An</i>	exception thrown from <code>_Alloc::allocate</code> or from the constructor of <code>_Tp</code> .
-----------	---

A copy of `__a` will be used to allocate memory for the `shared_ptr` and the new object.

Definition at line 857 of file `bits/shared_ptr.h`.

2.61.3.2 `atomic_compare_exchange_strong_explicit()`

```
template<typename _Tp >
bool atomic_compare_exchange_strong_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > * __v,
    shared_ptr< _Tp > __w,
    memory_order ,
    memory_order ) [related]
```

Atomic compare-and-swap for `shared_ptr` objects.

Parameters

<code>__p</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__v</code>	A non-null pointer to a <code>shared_ptr</code> object.
<code>__w</code>	A non-null pointer to a <code>shared_ptr</code> object.

Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`, or stronger than the memory order for success.

Definition at line 225 of file `shared_ptr_atomic.h`.

2.61.3.3 `atomic_exchange_explicit()`

```
template<typename _Tp >
shared_ptr< _Tp > atomic_exchange_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r,
    memory_order ) [related]
```

Atomic exchange for `shared_ptr` objects.

Parameters

$_p$	A non-null pointer to a <code>shared_ptr</code> object.
$_r$	New value to store in <code>*__p</code> .

Returns

The original value of `*__p`

Definition at line 176 of file `shared_ptr_atomic.h`.

2.61.3.4 `atomic_is_lock_free()`

```
template<typename _Tp , _Lock_policy _Lp>
bool atomic_is_lock_free (
    const __shared_ptr< _Tp, _Lp > * __p ) [related]
```

Report whether `shared_ptr` atomic operations are lock-free.

Parameters

$_p$	A non-null pointer to a <code>shared_ptr</code> object.
-------	---

Returns

True if atomic access to `*__p` is lock-free, false otherwise.

Definition at line 76 of file `shared_ptr_atomic.h`.

2.61.3.5 `atomic_load_explicit()`

```
template<typename _Tp >
shared_ptr< _Tp > atomic_load_explicit (
    const shared_ptr< _Tp > * __p,
    memory_order ) [related]
```

Atomic load for `shared_ptr` objects.

Parameters

\leftarrow <code>_p</code>	A non-null <code>shared_ptr</code> object.
---------------------------------	--

Returns

`*__p`

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

Definition at line 103 of file `shared_ptr_atomic.h`.

2.61.3.6 `atomic_store_explicit()`

```
template<typename _Tp >
void atomic_store_explicit (
    shared_ptr< _Tp > * __p,
    shared_ptr< _Tp > __r,
    memory_order ) [related]
```

Atomic store for `shared_ptr` objects.

Parameters

\leftarrow <code>_p</code>	A non-null pointer to a <code>shared_ptr</code> object.
\leftarrow <code>_r</code>	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

Definition at line 139 of file `shared_ptr_atomic.h`.

2.61.3.7 `const_pointer_cast()`

```
template<typename _Tp , typename _Up >
shared_ptr< _Tp > const_pointer_cast (
    const shared_ptr< _Up > & __r ) [related]
```

Convert type of `shared_ptr`, via `const_cast`

Definition at line 590 of file `bits/shared_ptr.h`.

2.61.3.8 `dynamic_pointer_cast()`

```
template<typename _Tp , typename _Up >
shared_ptr< _Tp > dynamic_pointer_cast (
    const shared_ptr< _Up > & __r ) [related]
```

Convert type of `shared_ptr`, via `dynamic_cast`

Definition at line 599 of file `bits/shared_ptr.h`.

2.61.3.9 `get_deleter()`

```
template<typename _Del , typename _Tp >
_Del * get_deleter (
    const shared_ptr< _Tp > & __p ) [related]
```

20.7.2.2.10 `shared_ptr` `get_deleter`

If `__p` has a deleter of type `_Del`, return a pointer to it.

Definition at line 93 of file `bits/shared_ptr.h`.

2.61.3.10 `make_shared()`

```
template<typename _Tp , typename... _Args>
shared_ptr< _Tp > make_shared (
    _Args &&... __args ) [related]
```

Create an object that is owned by a `shared_ptr`.

Parameters

<code>__args</code>	Arguments for the <code>_Tp</code> object's constructor.
---------------------	--

Returns

A `shared_ptr` that owns the newly created object.

Exceptions

<code>std::bad_alloc</code> , or	an exception thrown from the constructor of <code>_Tp</code> .
----------------------------------	--

Definition at line 872 of file `bits/shared_ptr.h`.

2.61.3.11 make_unique() [1/3]

```
template<typename _Tp , typename... _Args>
_MakeUniq< _Tp >::__single_object make_unique (
    _Args &&... __args ) [related]
```

std::make_unique for single objects

Definition at line 961 of file unique_ptr.h.

2.61.3.12 make_unique() [2/3]

```
template<typename _Tp >
_MakeUniq< _Tp >::__array make_unique (
    size_t __num ) [related]
```

std::make_unique for arrays of unknown bound

Definition at line 967 of file unique_ptr.h.

2.61.3.13 make_unique() [3/3]

```
template<typename _Tp , typename... _Args>
_MakeUniq< _Tp >::__invalid_type make_unique (
    _Args && ... ) [related]
```

Disable std::make_unique for arrays of known bound.

2.61.3.14 operator!=(()) [1/6]

```
template<typename _Tp , typename _Up >
bool operator!= (
    const shared_ptr< _Tp > & __a,
    const shared_ptr< _Up > & __b ) [related]
```

Inequality operator for shared_ptr objects, compares the stored pointers.

Definition at line 469 of file bits/shared_ptr.h.

2.61.3.15 operator!=(()) [2/6]

```
template<typename _Tp >
bool operator!=(
    const shared_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

shared_ptr comparison with nullptr

Definition at line 475 of file bits/shared_ptr.h.

2.61.3.16 operator!=(()) [3/6]

```
template<typename _Tp >
bool operator!=(
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 481 of file bits/shared_ptr.h.

2.61.3.17 operator!=(()) [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator!=(
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Inequality operator for unique_ptr objects, compares the owned pointers.

Definition at line 774 of file unique_ptr.h.

2.61.3.18 operator!=(()) [5/6]

```
template<typename _Tp , typename _Dp >
bool operator!=(
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

unique_ptr comparison with nullptr

Definition at line 781 of file unique_ptr.h.

2.61.3.19 operator!=(()) [6/6]

```
template<typename _Tp , typename _Dp >
bool operator!= (
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x ) [related]
```

unique_ptr comparison with nullptr

Definition at line 787 of file unique_ptr.h.

2.61.3.20 operator<() [1/6]

```
template<typename _Tp , typename _Up >
bool operator< (
    const shared_ptr< _Tp > & __a,
    const shared_ptr< _Up > & __b ) [related]
```

Relational operator for shared_ptr objects, compares the stored pointers.

Definition at line 487 of file bits/shared_ptr.h.

2.61.3.21 operator<() [2/6]

```
template<typename _Tp >
bool operator< (
    const shared_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

shared_ptr comparison with nullptr

Definition at line 498 of file bits/shared_ptr.h.

2.61.3.22 operator<() [3/6]

```
template<typename _Tp >
bool operator< (
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 507 of file bits/shared_ptr.h.

2.61.3.23 operator<>() [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator< (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Relational operator for unique_ptr objects, compares the owned pointers.

Definition at line 795 of file unique_ptr.h.

2.61.3.24 operator<>() [5/6]

```
template<typename _Tp , typename _Dp >
bool operator< (
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

unique_ptr comparison with nullptr

Definition at line 807 of file unique_ptr.h.

2.61.3.25 operator<>() [6/6]

```
template<typename _Tp , typename _Dp >
bool operator< (
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x ) [related]
```

unique_ptr comparison with nullptr

Definition at line 816 of file unique_ptr.h.

2.61.3.26 operator<<()

```
template<typename _Ch , typename _Tr , typename _Tp , _Lock_policy _Lp>
std::basic_ostream< _Ch, _Tr > & operator<< (
    std::basic_ostream< _Ch, _Tr > & __os,
    const __shared_ptr< _Tp, _Lp > & __p ) [related]
```

Write the stored pointer to an ostream.

Definition at line 69 of file bits/shared_ptr.h.

2.61.3.27 operator<=() [1/6]

```
template<typename _Tp , typename _Up >
bool operator<= (
    const shared_ptr< _Tp > & __a,
    const shared_ptr< _Up > & __b ) [related]
```

Relational operator for shared_ptr objects, compares the stored pointers.

Definition at line 516 of file bits/shared_ptr.h.

2.61.3.28 operator<=() [2/6]

```
template<typename _Tp >
bool operator<= (
    const shared_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

shared_ptr comparison with nullptr

Definition at line 522 of file bits/shared_ptr.h.

2.61.3.29 operator<=() [3/6]

```
template<typename _Tp >
bool operator<= (
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 528 of file bits/shared_ptr.h.

2.61.3.30 operator<=() [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator<= (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Relational operator for unique_ptr objects, compares the owned pointers.

Definition at line 826 of file unique_ptr.h.

2.61.3.31 operator<=() [5/6]

```
template<typename _Tp , typename _Dp >
bool operator<= (
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

unique_ptr comparison with nullptr

Definition at line 833 of file unique_ptr.h.

2.61.3.32 operator<=() [6/6]

```
template<typename _Tp , typename _Dp >
bool operator<= (
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x ) [related]
```

unique_ptr comparison with nullptr

Definition at line 839 of file unique_ptr.h.

2.61.3.33 operator==() [1/6]

```
template<typename _Tp , typename _Up >
bool operator== (
    const shared_ptr< _Tp > & __a,
    const shared_ptr< _Up > & __b ) [related]
```

Equality operator for shared_ptr objects, compares the stored pointers.

Definition at line 436 of file bits/shared_ptr.h.

2.61.3.34 operator==() [2/6]

```
template<typename _Tp >
bool operator== (
    const shared_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

shared_ptr comparison with nullptr

Definition at line 442 of file bits/shared_ptr.h.

2.61.3.35 operator==() [3/6]

```
template<typename _Tp >
bool operator==(
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 463 of file bits/shared_ptr.h.

2.61.3.36 operator==() [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator==(
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Equality operator for unique_ptr objects, compares the owned pointers.

Definition at line 753 of file unique_ptr.h.

2.61.3.37 operator==() [5/6]

```
template<typename _Tp , typename _Dp >
bool operator==(
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

unique_ptr comparison with nullptr

Definition at line 760 of file unique_ptr.h.

2.61.3.38 operator==() [6/6]

```
template<typename _Tp , typename _Dp >
bool operator==(
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x ) [related]
```

unique_ptr comparison with nullptr

Definition at line 767 of file unique_ptr.h.

2.61.3.39 operator>() [1/6]

```
template<typename _Tp , typename _Up >
bool operator> (
    const shared_ptr< _Tp > & __a,
    const shared_ptr< _Up > & __b ) [related]
```

Relational operator for shared_ptr objects, compares the stored pointers.

Definition at line 534 of file bits/shared_ptr.h.

2.61.3.40 operator>() [2/6]

```
template<typename _Tp >
bool operator> (
    const shared_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

shared_ptr comparison with nullptr

Definition at line 540 of file bits/shared_ptr.h.

2.61.3.41 operator>() [3/6]

```
template<typename _Tp >
bool operator> (
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 546 of file bits/shared_ptr.h.

2.61.3.42 operator>() [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator> (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Relational operator for unique_ptr objects, compares the owned pointers.

Definition at line 846 of file unique_ptr.h.

2.61.3.43 operator>() [5/6]

```
template<typename _Tp , typename _Dp >
bool operator> (
    const unique\_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

[unique_ptr](#) comparison with [nullptr](#)

Definition at line 853 of file [unique_ptr.h](#).

2.61.3.44 operator>() [6/6]

```
template<typename _Tp , typename _Dp >
bool operator> (
    nullptr_t ,
    const unique\_ptr< _Tp, _Dp > & __x ) [related]
```

[unique_ptr](#) comparison with [nullptr](#)

Definition at line 862 of file [unique_ptr.h](#).

2.61.3.45 operator>=() [1/6]

```
template<typename _Tp , typename _Up >
bool operator>= (
    const shared\_ptr< _Tp > & __a,
    const shared\_ptr< _Up > & __b ) [related]
```

Relational operator for [shared_ptr](#) objects, compares the stored pointers.

Definition at line 552 of file [bits/shared_ptr.h](#).

2.61.3.46 operator>=() [2/6]

```
template<typename _Tp >
bool operator>= (
    const shared\_ptr< _Tp > & __a,
    nullptr_t ) [related]
```

[shared_ptr](#) comparison with [nullptr](#)

Definition at line 558 of file [bits/shared_ptr.h](#).

2.61.3.47 operator>=() [3/6]

```
template<typename _Tp >
bool operator>= (
    nullptr_t ,
    const shared_ptr< _Tp > & __a ) [related]
```

shared_ptr comparison with nullptr

Definition at line 564 of file bits/shared_ptr.h.

2.61.3.48 operator>=() [4/6]

```
template<typename _Tp , typename _Dp , typename _Up , typename _Ep >
bool operator>= (
    const unique_ptr< _Tp, _Dp > & __x,
    const unique_ptr< _Up, _Ep > & __y ) [related]
```

Relational operator for unique_ptr objects, compares the owned pointers.

Definition at line 872 of file unique_ptr.h.

2.61.3.49 operator>=() [5/6]

```
template<typename _Tp , typename _Dp >
bool operator>= (
    const unique_ptr< _Tp, _Dp > & __x,
    nullptr_t ) [related]
```

unique_ptr comparison with nullptr

Definition at line 879 of file unique_ptr.h.

2.61.3.50 operator>=() [6/6]

```
template<typename _Tp , typename _Dp >
bool operator>= (
    nullptr_t ,
    const unique_ptr< _Tp, _Dp > & __x ) [related]
```

unique_ptr comparison with nullptr

Definition at line 885 of file unique_ptr.h.

2.61.3.51 static_pointer_cast()

```
template<typename _Tp , typename _Up >
shared_ptr< _Tp > static_pointer_cast (
    const shared_ptr< _Up > & __r ) [related]
```

Convert type of shared_ptr, via static_cast

Definition at line 581 of file bits/shared_ptr.h.

2.61.3.52 swap() [1/3]

```
template<typename _Tp >
void swap (
    shared_ptr< _Tp > & __a,
    shared_ptr< _Tp > & __b ) [related]
```

Swap overload for shared_ptr.

Definition at line 573 of file bits/shared_ptr.h.

Referenced by std::shared_ptr< _State >::atomic_exchange_explicit(), std::shared_ptr< _State >::atomic_store_explicit(), std::shared_ptr< _State >::swap(), and std::weak_ptr< _State_baseV2 >::swap().

2.61.3.53 swap() [2/3]

```
template<typename _Tp , typename _Dp >
enable_if< __is_swappable< _Dp >::value >::type swap (
    unique_ptr< _Tp, _Dp > & __x,
    unique_ptr< _Tp, _Dp > & __y ) [related]
```

Swap overload for unique_ptr.

Definition at line 738 of file unique_ptr.h.

2.61.3.54 swap() [3/3]

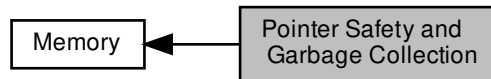
```
template<typename _Tp >
void swap (
    weak_ptr< _Tp > & __a,
    weak_ptr< _Tp > & __b ) [related]
```

Swap overload for weak_ptr.

Definition at line 762 of file bits/shared_ptr.h.

2.62 Pointer Safety and Garbage Collection

Collaboration diagram for Pointer Safety and Garbage Collection:



Enumerations

- enum `std::pointer_safety` { **relaxed**, **preferred**, **strict** }

Functions

- void `std::declare_no_pointers` (char *, size_t)
- void `std::declare_reachable` (void *)
- `pointer_safety` `std::get_pointer_safety` () noexcept
- void `std::undeclare_no_pointers` (char *, size_t)
- template<typename _Tp >
_Tp * `std::undeclare_reachable` (_Tp *__p)

2.62.1 Detailed Description

Utilities to assist with garbage collection in an implementation that supports *strict pointer safety*. This implementation only supports *relaxed pointer safety* and so these functions have no effect.

C++11 20.6.4 [util.dynamic.safety], Pointer safety

2.62.2 Enumeration Type Documentation

2.62.2.1 pointer_safety

```
enum std::pointer_safety [strong]
```

Constants representing the different types of pointer safety.

Definition at line 158 of file memory.

2.62.3 Function Documentation

2.62.3.1 `declare_no_pointers()`

```
void std::declare_no_pointers (
    char * ,
    size_t ) [inline]
```

Inform a garbage collector that a region of memory need not be traced.

Definition at line 171 of file memory.

2.62.3.2 `declare_reachable()`

```
void std::declare_reachable (
    void * ) [inline]
```

Inform a garbage collector that an object is still in use.

Definition at line 162 of file memory.

2.62.3.3 `get_pointer_safety()`

```
pointer_safety std::get_pointer_safety ( ) [inline], [noexcept]
```

The type of pointer safety supported by the implementation.

Definition at line 179 of file memory.

2.62.3.4 `undeclare_no_pointers()`

```
void std::undeclare_no_pointers (
    char * ,
    size_t ) [inline]
```

Unregister a range previously registered with `declare_no_pointers`.

Definition at line 175 of file memory.

2.62.3.5 `undeclare_reachable()`

```
template<typename _Tp >
_Tp* std::undeclare_reachable (
    _Tp * __p ) [inline]
```

Unregister an object previously registered with `declare_reachable`.

Definition at line 167 of file memory.

2.63 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class `std::discrete_distribution< _IntType >`
- class `std::exponential_distribution< _RealType >`
- class `std::extreme_value_distribution< _RealType >`
- struct `std::piecewise_linear_distribution< _RealType >::param_type`
- struct `std::piecewise_constant_distribution< _RealType >::param_type`
- struct `std::discrete_distribution< _IntType >::param_type`
- struct `std::extreme_value_distribution< _RealType >::param_type`
- struct `std::weibull_distribution< _RealType >::param_type`
- struct `std::exponential_distribution< _RealType >::param_type`
- struct `std::poisson_distribution< _IntType >::param_type`
- class `std::piecewise_constant_distribution< _RealType >`
- class `std::piecewise_linear_distribution< _RealType >`
- class `std::poisson_distribution< _IntType >`
- class `std::weibull_distribution< _RealType >`

Functions

- template<typename _IntType >
bool `std::operator!=` (const `std::poisson_distribution< _IntType >` &__d1, const `std::poisson_distribution< _IntType >` &__d2)
- template<typename _RealType >
bool `std::operator!=` (const `std::exponential_distribution< _RealType >` &__d1, const `std::exponential_distribution< _RealType >` &__d2)
- template<typename _RealType >
bool `std::operator!=` (const `std::weibull_distribution< _RealType >` &__d1, const `std::weibull_distribution< _RealType >` &__d2)
- template<typename _RealType >
bool `std::operator!=` (const `std::extreme_value_distribution< _RealType >` &__d1, const `std::extreme_value_distribution< _RealType >` &__d2)
- template<typename _IntType >
bool `std::operator!=` (const `std::discrete_distribution< _IntType >` &__d1, const `std::discrete_distribution< _IntType >` &__d2)

- `template<typename _RealType >`
`bool std::operator!=(const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`
`bool std::operator!=(const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`

2.63.1 Detailed Description

2.63.2 Function Documentation

2.63.2.1 `operator!=()` [1/7]

```
template<typename _IntType >
bool std::operator!=(
    const std::poisson_distribution< _IntType > & __d1,
    const std::poisson_distribution< _IntType > & __d2 ) [inline]
```

Return true if two Poisson distributions are different.

Definition at line 4624 of file random.h.

2.63.2.2 `operator!=()` [2/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::exponential_distribution< _RealType > & __d1,
    const std::exponential_distribution< _RealType > & __d2 ) [inline]
```

Return true if two exponential distributions have different parameters.

Definition at line 4815 of file random.h.

2.63.2.3 operator!=() [3/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::weibull_distribution< _RealType > & __d1,
    const std::weibull_distribution< _RealType > & __d2 ) [inline]
```

Return true if two Weibull distributions have different parameters.

Definition at line 5025 of file random.h.

2.63.2.4 operator!=() [4/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::extreme_value_distribution< _RealType > & __d1,
    const std::extreme_value_distribution< _RealType > & __d2 ) [inline]
```

Return true if two extreme value distributions have different parameters.

Definition at line 5235 of file random.h.

2.63.2.5 operator!=() [5/7]

```
template<typename _IntType >
bool std::operator!=(
    const std::discrete_distribution< _IntType > & __d1,
    const std::discrete_distribution< _IntType > & __d2 ) [inline]
```

Return true if two discrete distributions have different parameters.

Definition at line 5500 of file random.h.

2.63.2.6 operator!=() [6/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::piecewise_constant_distribution< _RealType > & __d1,
    const std::piecewise_constant_distribution< _RealType > & __d2 ) [inline]
```

Return true if two piecewise constant distributions have different parameters.

Definition at line 5771 of file random.h.

2.63.2.7 operator!=() [7/7]

```
template<typename _RealType >
bool std::operator!=(
    const std::piecewise_linear_distribution< _RealType > & __d1,
    const std::piecewise_linear_distribution< _RealType > & __d2 ) [inline]
```

Return true if two piecewise linear distributions have different parameters.

Definition at line 6044 of file random.h.

2.63.2.8 operator<<() [1/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<(
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::exponential_distribution< _RealType > & __x )
```

Inserts a exponential_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A exponential_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 1719 of file bits/random.tcc.

References std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), and std::ios_base::precision().

2.63.2.9 operator<<() [2/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<<(
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::weibull_distribution< _RealType > & __x )
```

Inserts a weibull_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A weibull_distribution random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2504 of file `bits/random.tcc`.

References `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::precision()`, and `std::basic_istream<_CharT, _Traits>::widen()`.

2.63.2.10 operator<<() [3/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::extreme_value_distribution< _RealType > & __x )
```

Inserts a `extreme_value_distribution` random number distribution `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number distribution.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 2578 of file `bits/random.tcc`.

References `std::basic_ios<_CharT, _Traits>::fill()`, `std::ios_base::flags()`, `std::ios_base::precision()`, and `std::basic_istream<_CharT, _Traits>::widen()`.

2.63.2.11 operator>>() [1/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::exponential_distribution< _RealType > & __x )
```

Extracts a `exponential_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>exponential_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 1741 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::exponential_distribution<_RealType>::param()`, and `std::skipws()`.

2.63.2.12 operator>>() [2/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::weibull_distribution< _RealType > & __x )
```

Extracts a `weibull_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>weibull_distribution</code> random number generator engine.

Returns

The input stream with `__x` extracted or in an error state.

Definition at line 2527 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::weibull_distribution<_RealType>::param()`, and `std::skipws()`.

2.63.2.13 operator>>() [3/3]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::extreme_value_distribution< _RealType > & __x )
```

Extracts a `extreme_value_distribution` random number distribution `__x` from the input stream `__is`.

Parameters

<code>__is</code>	An input stream.
<code>__x</code>	A <code>extreme_value_distribution</code> random number generator engine.

Returns

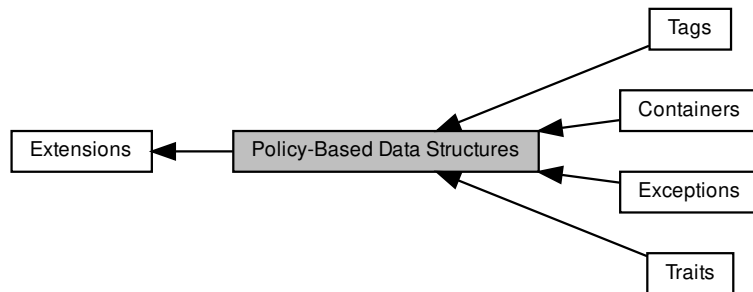
The input stream with `__x` extracted or in an error state.

Definition at line 2601 of file `bits/random.tcc`.

References `std::dec()`, `std::ios_base::flags()`, `std::extreme_value_distribution<_RealType>::param()`, and `std::skipws()`.

2.64 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Modules

- [Containers](#)
- [Exceptions](#)
- [Tags](#)
- [Traits](#)

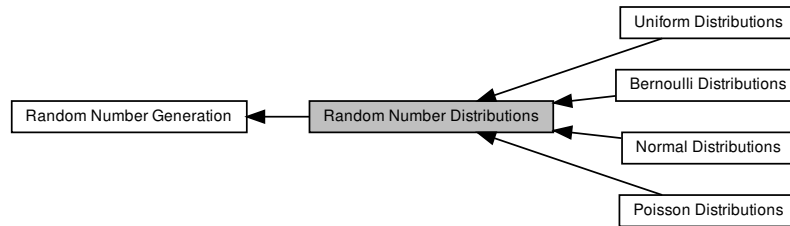
2.64.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

2.65 Random Number Distributions

Collaboration diagram for Random Number Distributions:



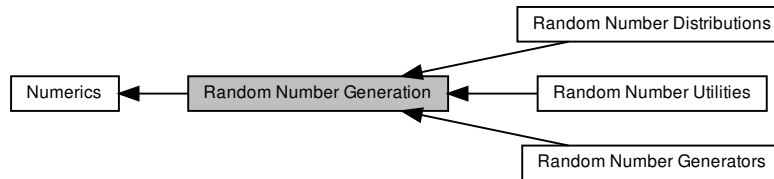
Modules

- [Bernoulli Distributions](#)
- [Normal Distributions](#)
- [Poisson Distributions](#)
- [Uniform Distributions](#)

2.65.1 Detailed Description

2.66 Random Number Generation

Collaboration diagram for Random Number Generation:



Modules

- [Random Number Distributions](#)
- [Random Number Generators](#)
- [Random Number Utilities](#)

Namespaces

- [std::__detail](#)

Functions

- `template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator > _RealType std::generate_canonical (_UniformRandomNumberGenerator & __g)`

2.66.1 Detailed Description

A facility for generating random numbers on selected distributions.

2.66.2 Function Documentation

2.66.2.1 `generate_canonical()`

```

template<typename _RealType , size_t __bits, typename _UniformRandomNumberGenerator >
_RealType std::generate_canonical (
    _UniformRandomNumberGenerator & __g )
  
```

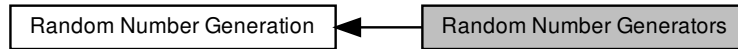
A function template for converting the output of a (integral) uniform random number generator to a floating point result in the range [0-1).

Definition at line 3282 of file `bits/random.tcc`.

References `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::min()`.

2.67 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`
- class `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::random_device`
- class `std::shuffle_order_engine< _RandomNumberEngine, __k >`
- class `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`

Typedefs

- typedef `minstd_rand0 std::default_random_engine`
- typedef `shuffle_order_engine< minstd_rand0, 256 > std::knuth_b`
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL > std::mt19937_64`
- typedef `discard_block_engine< ranlux24_base, 223, 23 > std::ranlux24`
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > std::ranlux24_base`
- typedef `discard_block_engine< ranlux48_base, 389, 11 > std::ranlux48`
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > std::ranlux48_base`

Functions

- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool `std::operator!=` (const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__lhs, const `std::linear_congruential_engine< _UIntType, __a, __c, __m >` &__rhs)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool `std::operator!=` (const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &__lhs, const `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >` &__rhs)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const`
`std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const`
`std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs,`
`const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const`
`std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`

2.67.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.

A number generator is a function object with an `operator()` that takes zero arguments and returns a number.

A compliant random number generator must satisfy the following requirements.

Table 229 Random Number Generator Requirements

To be documented.

2.67.2 Typedef Documentation

2.67.2.1 minstd_rand

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

Definition at line 1560 of file `random.h`.

2.67.2.2 minstd_rand0

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard `rand0` of Lewis, Goodman, and Miller.

Definition at line 1554 of file `random.h`.

2.67.2.3 mt19937

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL,
7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.

Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

Definition at line 1576 of file random.h.

2.67.2.4 mt19937_64

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
0x5555555555555555ULL, 17, 0x71d67fffed60000ULL, 37, 0xffff7eee00000000ULL, 43, 6364136223846793005ULL> std::mt19937_64
```

An alternative Mersenne Twister.

Definition at line 1588 of file random.h.

2.67.3 Function Documentation

2.67.3.1 operator!=() [1/6]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool std::operator!=(
    const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
    const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs ) [inline]
```

Compares two linear congruential random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A linear congruential random number generator object.
<code>__rhs</code>	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 432 of file random.h.

2.67.3.2 operator!=() [2/6]

```
template<typename _UIntType , size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a,
size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _U
UIntType __f>
bool std::operator!=( (
    const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __
__s, __b, __t, __c, __l, __f > & __lhs,
    const std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __
__s, __b, __t, __c, __l, __f > & __rhs ) [inline]
```

Compares two % mersenne_twister_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % mersenne_twister_engine random number generator object.
<code>__rhs</code>	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 669 of file random.h.

2.67.3.3 operator!=() [3/6]

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
bool std::operator!=( (
    const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
    const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs ) [inline]
```

Compares two % subtract_with_carry_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A % subtract_with_carry_engine random number generator object.
<code>__rhs</code>	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 870 of file random.h.

2.67.3.4 `operator!=()` [4/6]

```
template<typename _RandomNumberEngine , size_t __p, size_t __r>
bool std::operator!= (
    const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
    const std::discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs ) [inline]
```

Compares two `discard_block_engine` random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A <code>discard_block_engine</code> random number generator object.
<code>__rhs</code>	Another <code>discard_block_engine</code> random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1093 of file `random.h`.

2.67.3.5 `operator!=()` [5/6]

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
bool std::operator!= (
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs )
[inline]
```

Compares two `independent_bits_engine` random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A <code>independent_bits_engine</code> random number generator object.
<code>__rhs</code>	Another <code>independent_bits_engine</code> random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1290 of file `random.h`.

2.67.3.6 operator!=() [6 / 6]

```
template<typename _RandomNumberEngine , size_t __k>
bool std::operator!=(
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
    const std::shuffle_order_engine< _RandomNumberEngine, __k > & __rhs ) [inline]
```

Compares two shuffle_order_engine random number generator objects of the same type for inequality.

Parameters

<code>__lhs</code>	A shuffle_order_engine random number generator object.
<code>__rhs</code>	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be different, false otherwise.

Definition at line 1543 of file random.h.

2.67.3.7 operator<<()

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType , typename _CharT , typename
 Traits >
std::basic_ostream<_CharT, Traits>& std::operator<< (
    std::basic_ostream< _CharT, Traits > & __os,
    const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x )
```

Inserts the current state of a independent_bits_engine random number generator engine `__x` into the output stream `__os`.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A independent_bits_engine random number generator engine.

Returns

The output stream with the state of `__x` inserted or in an error state.

Definition at line 1309 of file random.h.

2.68 Random Number Utilities

Collaboration diagram for Random Number Utilities:



Classes

- class `std::seed_seq`

2.68.1 Detailed Description

2.69 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Files

- file [ratio](#)

Classes

- struct [std::ratio< _Num, _Den >](#)
- struct [std::ratio_equal< _R1, _R2 >](#)
- struct [std::ratio_greater< _R1, _R2 >](#)
- struct [std::ratio_greater_equal< _R1, _R2 >](#)
- struct [std::ratio_less< _R1, _R2 >](#)
- struct [std::ratio_less_equal< _R1, _R2 >](#)
- struct [std::ratio_not_equal< _R1, _R2 >](#)

Typedefs

- typedef [ratio](#)< 1, 1000000000000000000 > **std::atto**
- typedef [ratio](#)< 1, 100 > **std::centi**
- typedef [ratio](#)< 10, 1 > **std::deca**
- typedef [ratio](#)< 1, 10 > **std::dec**
- typedef [ratio](#)< 1000000000000000000, 1 > **std::exa**
- typedef [ratio](#)< 1, 1000000000000000000 > **std::femto**
- typedef [ratio](#)< 1000000000, 1 > **std::giga**
- typedef [ratio](#)< 100, 1 > **std::hecto**
- typedef [ratio](#)< 1000, 1 > **std::kilo**
- typedef [ratio](#)< 1000000, 1 > **std::mega**
- typedef [ratio](#)< 1, 1000000 > **std::micro**
- typedef [ratio](#)< 1, 1000 > **std::milli**
- typedef [ratio](#)< 1, 1000000000 > **std::nano**
- typedef [ratio](#)< 1000000000000000000, 1 > **std::peta**
- typedef [ratio](#)< 1, 1000000000000000 > **std::pico**
- template<typename _R1, typename _R2 >
using [std::ratio_add](#) = typename __ratio_add< _R1, _R2 >::type
- template<typename _R1, typename _R2 >
using [std::ratio_divide](#) = typename __ratio_divide< _R1, _R2 >::type
- template<typename _R1, typename _R2 >
using [std::ratio_multiply](#) = typename __ratio_multiply< _R1, _R2 >::type
- template<typename _R1, typename _R2 >
using [std::ratio_subtract](#) = typename __ratio_subtract< _R1, _R2 >::type
- typedef [ratio](#)< 1000000000000000, 1 > **std::tera**

Variables

- static constexpr intmax_t **std::ratio**<_Num, _Den>::den
- static constexpr intmax_t **std::ratio**<_Num, _Den>::num

2.69.1 Detailed Description

Compile time representation of finite rational numbers.

2.69.2 Typedef Documentation

2.69.2.1 ratio_add

```
template<typename _R1 , typename _R2 >  
using std::ratio_add = typedef typename __ratio_add<_R1, _R2>::type
```

ratio_add

Definition at line 525 of file ratio.

2.69.2.2 ratio_divide

```
template<typename _R1 , typename _R2 >  
using std::ratio_divide = typedef typename __ratio_divide<_R1, _R2>::type
```

ratio_divide

Definition at line 347 of file ratio.

2.69.2.3 ratio_multiply

```
template<typename _R1 , typename _R2 >  
using std::ratio_multiply = typedef typename __ratio_multiply<_R1, _R2>::type
```

ratio_multiply

Definition at line 320 of file ratio.

2.69.2.4 ratio_subtract

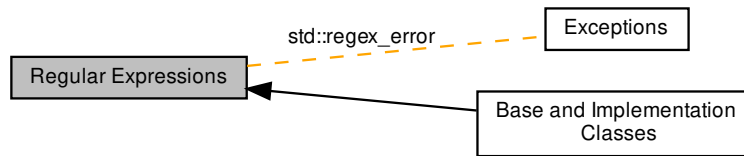
```
template<typename _R1 , typename _R2 >  
using std::ratio_subtract = typedef typename __ratio_subtract<_R1, _R2>::type
```

ratio_subtract

Definition at line 550 of file ratio.

2.70 Regular Expressions

Collaboration diagram for Regular Expressions:



Modules

- [Base and Implementation Classes](#)

Namespaces

- [std::regex_constants](#)

Classes

- class [std::basic_regex< _Ch_type, _Rx_traits >](#)
- class [std::match_results< _Bi_iter, _Alloc >](#)
- class [std::regex_error](#)
- class [std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- class [std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >](#)
- class [std::regex_traits< _Ch_type >](#)
- class [std::sub_match< _Biter >](#)

Typedefs

- typedef [match_results< const char * >](#) **std::cmatch**
- typedef [regex_iterator< const char * >](#) **std::cregex_iterator**
- typedef [regex_token_iterator< const char * >](#) **std::cregex_token_iterator**
- typedef [sub_match< const char * >](#) **std::csub_match**
- typedef [basic_regex< char >](#) **std::regex**
- typedef [match_results< string::const_iterator >](#) **std::smatch**
- typedef [regex_iterator< string::const_iterator >](#) **std::sregex_iterator**
- typedef [regex_token_iterator< string::const_iterator >](#) **std::sregex_token_iterator**
- typedef [sub_match< string::const_iterator >](#) **std::ssub_match**
- typedef [match_results< const wchar_t * >](#) **std::wcmatch**
- typedef [regex_iterator< const wchar_t * >](#) **std::wcregex_iterator**
- typedef [regex_token_iterator< const wchar_t * >](#) **std::wcregex_token_iterator**
- typedef [sub_match< const wchar_t * >](#) **std::wcs_sub_match**
- typedef [basic_regex< wchar_t >](#) **std::wregex**
- typedef [match_results< wstring::const_iterator >](#) **std::wsmatch**
- typedef [regex_iterator< wstring::const_iterator >](#) **std::wsregex_iterator**
- typedef [regex_token_iterator< wstring::const_iterator >](#) **std::wsregex_token_iterator**
- typedef [sub_match< wstring::const_iterator >](#) **std::wssub_match**

Functions

- `template<typename _Bi_iter, class _Alloc >`
`bool std::operator!= (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Bi_iter, typename _Alloc >`
`bool std::operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _Ch_type, typename _Rx_traits >`
`void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs)`
- `template<typename _Bi_iter, typename _Alloc >`
`void std::swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept`

- `template<typename _Bilter >`
`bool operator== (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter >`
`bool operator!= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter >`
`bool operator< (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter >`
`bool operator<= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter >`
`bool operator>= (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bilter >`
`bool operator> (const sub_match< _Bilter > &__lhs, const sub_match< _Bilter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator< (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator> (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator>= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator<= (const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`
- `template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > &__rhs)`

- [illegible]

- `template<typename _Bi_iter >`
`bool operator> (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (typename iterator_traits< _Bi_iter >::value_type const &__lhs, const sub_match< _Bi_iter > &__rhs)`
- `template<typename _Bi_iter >`
`bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator!= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator< (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator>= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Bi_iter >`
`bool operator<= (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)`
- `template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter >`
`basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)`

Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type __f=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits >`
`bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`

- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex<`
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e,`
`regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type,`
`_Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`
`_Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__`
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`
`basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx`
`_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`
`basic_string< _Ch_type > std::regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx`
`_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

Constants

std [28.8.1](1)

- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::icase`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::nosubs`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::optimize`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::collate`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::ECMAScript`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::basic`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::extended`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::awk`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::grep`
- static constexpr `flag_type` `std::basic_regex<_Ch_type, _Rx_traits>::egrep`

2.70.1 Detailed Description

A facility for performing regular expression pattern matching.

2.70.2 Typedef Documentation

2.70.2.1 `cregex_token_iterator`

```
typedef regex_token_iterator<const char*> std::cregex_token_iterator
```

Token iterator for C-style NULL-terminated strings.

Definition at line 2963 of file `regex.h`.

2.70.2.2 `sub_match`

```
typedef sub_match<const char*> std::sub_match
```

Standard regex submatch over a C-style null-terminated string.

Definition at line 1010 of file `regex.h`.

2.70.2.3 `regex`

```
typedef basic_regex<char> std::regex
```

Standard regular expressions.

Definition at line 832 of file `regex.h`.

2.70.2.4 `sregex_token_iterator`

```
typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator
```

Token iterator for standard strings.

Definition at line 2966 of file `regex.h`.

2.70.2.5 `ssub_match`

```
typedef sub_match<string::const_iterator> std::ssub_match
```

Standard regex submatch over a standard string.

Definition at line 1013 of file `regex.h`.

2.70.2.6 `wcregex_token_iterator`

```
typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator
```

Token iterator for C-style NULL-terminated wide strings.

Definition at line 2970 of file `regex.h`.

2.70.2.7 `wcsub_match`

```
typedef sub_match<const wchar_t*> std::wcsub_match
```

Regex submatch over a C-style null-terminated wide string.

Definition at line 1017 of file `regex.h`.

2.70.2.8 `wregex`

```
typedef basic_regex<wchar_t> std::wregex
```

Standard wide-character regular expressions.

Definition at line 836 of file `regex.h`.

2.70.2.9 wsregex_token_iterator

```
typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator
```

Token iterator for standard wide-character strings.

Definition at line 2973 of file regex.h.

2.70.2.10 wssub_match

```
typedef sub_match<wstring::const_iterator> std::wssub_match
```

Regex submatch over a standard wide string.

Definition at line 1020 of file regex.h.

2.70.3 Function Documentation

2.70.3.1 operator!=() [1/8]

```
template<typename _BiIter >  
bool operator!=(  
    const sub_match< _BiIter > & __lhs,  
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the inequivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1064 of file regex.h.

2.70.3.2 `operator!=()` [2/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator!= (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub\_match< _Bi_iter > & __rhs ) [related]
```

Tests the inequivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1144 of file `regex.h`.

2.70.3.3 `operator!=()` [3/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator!= (
    const sub\_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1237 of file `regex.h`.

2.70.3.4 `operator!=()` [4/8]

```
template<typename _Bi_iter >
bool operator!= (
```

```
typename iterator_traits<_Bi_iter>::value_type const * __lhs,  
const sub_match<_Bi_iter> & __rhs ) [related]
```

Tests the inequivalence of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1311 of file `regex.h`.

2.70.3.5 `operator!=()` [5/8]

```
template<typename _Bi_iter >
bool operator!= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the inequivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1405 of file `regex.h`.

2.70.3.6 `operator!=()` [6/8]

```
template<typename _Bi_iter >
bool operator!= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the inequivalence of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1479 of file `regex.h`.

2.70.3.7 operator!=(()) [7/8]

```
template<typename _Bi_iter >
bool operator!= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the inequivalence of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` is not equivalent to `__rhs`, false otherwise.

Definition at line 1579 of file `regex.h`.

2.70.3.8 operator!=(()) [8/8]

```
template<typename _Bi_iter , class _Alloc >
bool std::operator!= (
    const match_results< _Bi_iter, _Alloc > & __m1,
    const match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two `match_results` for inequality.

Returns

true if the two objects do not refer to the same match, false otherwise.

Definition at line 2126 of file `regex.h`.

2.70.3.9 operator<() [1/7]

```
template<typename _BiIter >
bool operator< (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1075 of file `regex.h`.

2.70.3.10 `operator<()` [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator< (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub\_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1156 of file `regex.h`.

2.70.3.11 `operator<()` [3/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator< (
    const sub\_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1249 of file `regex.h`.

2.70.3.12 operator<>() [4/7]

```
template<typename _Bi_iter >
bool operator< (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1323 of file `regex.h`.

2.70.3.13 operator<>() [5/7]

```
template<typename _Bi_iter >
bool operator< (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a C string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1417 of file regex.h.

2.70.3.14 `operator<()` [6/7]

```
template<typename _Bi_iter >
bool operator< (
    typename iterator\_traits< _Bi_iter >::value_type const & __lhs,
    const sub\_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1492 of file regex.h.

2.70.3.15 `operator<()` [7/7]

```
template<typename _Bi_iter >
bool operator< (
    const sub\_match< _Bi_iter > & __lhs,
    typename iterator\_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` precedes `__rhs`, false otherwise.

Definition at line 1592 of file regex.h.

2.70.3.16 `operator<<()`

```
template<typename _Ch_type , typename _Ch_traits , typename _Bi_iter >
basic_ostream< _Ch_type, _Ch_traits > & operator<< (
    basic_ostream< _Ch_type, _Ch_traits > & __os,
    const sub_match< _Bi_iter > & __m ) [related]
```

Inserts a matched string into an output stream.

Parameters

<code>__os</code>	The output stream.
<code>__m</code>	A submatch string.

Returns

the output stream with the submatch string inserted.

Definition at line 1647 of file regex.h.

2.70.3.17 `operator<=()` [1/7]

```
template<typename _BiIter >
bool operator<= (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1086 of file regex.h.

2.70.3.18 `operator<=()` [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator<= (
```

```
const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,  
const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1192 of file `regex.h`.

2.70.3.19 `operator<=()` [3/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator<= (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1285 of file `regex.h`.

2.70.3.20 `operator<=()` [4/7]

```
template<typename _Bi_iter >
bool operator<= (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1359 of file `regex.h`.

2.70.3.21 operator<=() [5/7]

```
template<typename _Bi_iter >
bool operator<= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a C string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1453 of file `regex.h`.

2.70.3.22 operator<=() [6/7]

```
template<typename _Bi_iter >
bool operator<= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1531 of file `regex.h`.

2.70.3.23 operator<=() [7/7]

```
template<typename _Bi_iter >
bool operator<= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` does not succeed `__rhs`, false otherwise.

Definition at line 1631 of file regex.h.

2.70.3.24 operator==() [1/8]

```
template<typename _BiIter >
bool operator== (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the equivalence of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1035 of file regex.h.

2.70.3.25 operator==() [2/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator== (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the equivalence of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1131 of file regex.h.

2.70.3.26 operator==() [3/8]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator== (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the equivalence of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1206 of file regex.h.

2.70.3.27 operator==() [4/8]

```
template<typename _Bi_iter >
bool operator== (
```



```
typename iterator_traits<_Bi_iter>::value_type const * __lhs,  
const sub_match<_Bi_iter> & __rhs ) [related]
```

Tests the equivalence of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1298 of file `regex.h`.

2.70.3.28 `operator==()` [5/8]

```
template<typename _Bi_iter >
bool operator== (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the equivalence of a regular expression submatch and a C string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1373 of file `regex.h`.

2.70.3.29 `operator==()` [6/8]

```
template<typename _Bi_iter >
bool operator== (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the equivalence of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1466 of file `regex.h`.

2.70.3.30 operator==() [7/8]

```
template<typename _Bi_iter >
bool operator== (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the equivalence of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` is equivalent to `__rhs`, false otherwise.

Definition at line 1545 of file `regex.h`.

2.70.3.31 operator==() [8/8]

```
template<typename _Bi_iter , typename _Alloc >
bool std::operator== (
    const match_results< _Bi_iter, _Alloc > & __m1,
    const match_results< _Bi_iter, _Alloc > & __m2 ) [inline]
```

Compares two `match_results` for equality.

Returns

true if the two objects refer to the same match, false otherwise.

Definition at line 2101 of file `regex.h`.

References `std::match_results< _Bi_iter, _Alloc >::begin()`, `std::match_results< _Bi_iter, _Alloc >::empty()`, `std::match_results< _Bi_iter, _Alloc >::end()`, `std::equal()`, `std::match_results< _Bi_iter, _Alloc >::prefix()`, `std::match_results< _Bi_iter, _Alloc >::ready()`, `std::match_results< _Bi_iter, _Alloc >::size()`, and `std::match_results< _Bi_iter, _Alloc >::suffix()`.

2.70.3.32 `operator>()` [1/7]

```
template<typename _BiIter >
bool operator> (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1108 of file `regex.h`.

2.70.3.33 `operator>()` [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator> (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1168 of file `regex.h`.

2.70.3.34 `operator>()` [3/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator> (
```

```
const sub_match< _Bi_iter > & __lhs,  
const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1261 of file `regex.h`.

2.70.3.35 `operator>()` [4/7]

```
template<typename _Bi_iter >
bool operator> (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1335 of file `regex.h`.

2.70.3.36 `operator>()` [5/7]

```
template<typename _Bi_iter >
bool operator> (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a C string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1429 of file `regex.h`.

2.70.3.37 operator>() [6/7]

```
template<typename _Bi_iter >
bool operator> (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1505 of file `regex.h`.

2.70.3.38 operator>() [7/7]

```
template<typename _Bi_iter >
bool operator> (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` succeeds `__rhs`, false otherwise.

Definition at line 1605 of file `regex.h`.

2.70.3.39 operator>=() [1/7]

```
template<typename _BiIter >
bool operator>= (
    const sub_match< _BiIter > & __lhs,
    const sub_match< _BiIter > & __rhs ) [related]
```

Tests the ordering of two regular expression submatches.

Parameters

<code>__lhs</code>	First regular expression submatch.
<code>__rhs</code>	Second regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1097 of file regex.h.

2.70.3.40 operator>=() [2/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator>= (
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a string and a regular expression submatch.

Parameters

<code>__lhs</code>	A string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1180 of file regex.h.

2.70.3.41 `operator>=()` [3/7]

```
template<typename _Bi_iter , typename _Ch_traits , typename _Ch_alloc >
bool operator>= (
    const sub_match< _Bi_iter > & __lhs,
    const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1273 of file `regex.h`.

2.70.3.42 `operator>=()` [4/7]

```
template<typename _Bi_iter >
bool operator>= (
    typename iterator_traits< _Bi_iter >::value_type const * __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a C string and a regular expression submatch.

Parameters

<code>__lhs</code>	A null-terminated string.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1347 of file `regex.h`.

2.70.3.43 `operator>=()` [5/7]

```
template<typename _Bi_iter >
bool operator>= (
```

```
const sub_match< _Bi_iter > & __lhs,  
typename iterator_traits< _Bi_iter >::value_type const * __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a C string.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A null-terminated string.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1441 of file `regex.h`.

2.70.3.44 `operator>=()` [6/7]

```
template<typename _Bi_iter >
bool operator>= (
    typename iterator_traits< _Bi_iter >::value_type const & __lhs,
    const sub_match< _Bi_iter > & __rhs ) [related]
```

Tests the ordering of a character and a regular expression submatch.

Parameters

<code>__lhs</code>	A character.
<code>__rhs</code>	A regular expression submatch.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1518 of file `regex.h`.

2.70.3.45 `operator>=()` [7/7]

```
template<typename _Bi_iter >
bool operator>= (
    const sub_match< _Bi_iter > & __lhs,
    typename iterator_traits< _Bi_iter >::value_type const & __rhs ) [related]
```

Tests the ordering of a regular expression submatch and a character.

Parameters

<code>__lhs</code>	A regular expression submatch.
<code>__rhs</code>	A character.

Returns

true if `__lhs` does not precede `__rhs`, false otherwise.

Definition at line 1618 of file `regex.h`.

2.70.3.46 `regex_match()` [1/7]

```
template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
bool std::regex_match (
    _Bi_iter __s,
    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<code>__s</code>	Start of the character sequence to match.
<code>__e</code>	One-past-the-end of the character sequence to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2171 of file `regex.h`.

Referenced by `std::regex_match()`.

2.70.3.47 `regex_match()` [2/7]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_match (
    _Bi_iter __first,
```

```

    __Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]

```

Indicates if there is a match between the regular expression `e` and all of the character sequence `[first, last)`.

Parameters

<code>__first</code>	Beginning of the character sequence to match.
<code>__last</code>	One-past-the-end of the character sequence to match.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2199 of file `regex.h`.

References `std::regex_match()`.

2.70.3.48 `regex_match()` [3/7]

```

template<typename _Ch_type , typename _Alloc , typename _Rx_traits >
bool std::regex_match (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]

```

Determines if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2224 of file `regex.h`.

References `std::regex_match()`.

2.70.3.49 `regex_match()` [4/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator,
    _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Determines if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	The string to match.
<code>__m</code>	The match results.
<code>__re</code>	The regular expression.
<code>__flags</code>	Controls how the regular expression is matched.

Return values

<i>true</i>	A match exists.
<i>false</i>	Otherwise.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2248 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

2.70.3.50 `regex_match()` [5/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & ,
    const basic_regex< _Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default ) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

2.70.3.51 `regex_match()` [6/7]

```
template<typename _Ch_type , class _Rx_traits >
bool std::regex_match (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a C-style null-terminated string.

Parameters

<code>__s</code>	The C-style null-terminated string to match.
<code>__re</code>	The regular expression.
<code>__f</code>	Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2284 of file `regex.h`.

References `std::regex_match()`.

2.70.3.52 `regex_match()` [7/7]

```
template<typename _Ch_traits , typename _Str_allocator , typename _Ch_type , typename _Rx_traits
>
bool std::regex_match (
    const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Indicates if there is a match between the regular expression `e` and a string.

Parameters

<code>__s</code>	[IN] The string to match.
<code>__re</code>	[IN] The regular expression.
<code>__flags</code>	[IN] Controls how the regular expression is matched.

Return values

<code>true</code>	A match exists.
<code>false</code>	Otherwise.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2306 of file `regex.h`.

References `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_match()`.

2.70.3.53 `regex_replace()` [1/6]

```
template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type , typename
_St , typename _Sa >
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const basic_string< _Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

`__out`

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2477 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

Referenced by `std::regex_replace()`.

2.70.3.54 `regex_replace()` [2/6]

```
template<typename _Out_iter , typename _Bi_iter , typename _Rx_traits , typename _Ch_type >
_Out_iter std::regex_replace (
    _Out_iter __out,
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex<_Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default )
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__out</code>	[OUT] The output iterator.
<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns`__out`**Exceptions**

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

2.70.3.55 `regex_replace()` [3/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa , typename _Fst ,
typename _Fsa >
basic_string<_Ch_type, _St, _Sa> std::regex_replace (
    const basic_string< _Ch_type, _St, _Sa > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2522 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

2.70.3.56 `regex_replace()` [4/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa >
basic_string<_Ch_type, _St, _Sa> std::regex_replace (
    const basic_string< _Ch_type, _St, _Sa > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__s</code>	[IN] The string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2548 of file `regex.h`.

References `std::back_inserter()`, `std::basic_string< _CharT, _Traits, _Alloc >::begin()`, `std::basic_string< _CharT, _Traits, _Alloc >::end()`, and `std::regex_replace()`.

2.70.3.57 `regex_replace()` [5/6]

```
template<typename _Rx_traits , typename _Ch_type , typename _St , typename _Sa >
basic_string<_Ch_type> std::regex_replace (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const basic_string< _Ch_type, _St, _Sa > & __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2574 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

2.70.3.58 `regex_replace()` [6/6]

```
template<typename _Rx_traits , typename _Ch_type >
basic_string<_Ch_type> std::regex_replace (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    const _Ch_type * __fmt,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

<code>__s</code>	[IN] The C-string to search and replace.
<code>__e</code>	[IN] The regular expression to search for.
<code>__fmt</code>	[IN] The format C-string.
<code>__flags</code>	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2600 of file `regex.h`.

References `std::back_inserter()`, and `std::regex_replace()`.

2.70.3.59 `regex_search()` [1/7]

```
template<typename _Bi_iter , typename _Alloc , typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    _Bi_iter __s,
    _Bi_iter __e,
    match_results< _Bi_iter, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

<code>__s</code>	[IN] The start of the string to search.
<code>__e</code>	[IN] One-past-the-end of the string to search.
<code>__m</code>	[OUT] The match results.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string, the content of <code>m</code> is undefined.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2329 of file `regex.h`.

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`, and `std::regex_search()`.

2.70.3.60 `regex_search()` [2/7]

```
template<typename _Bi_iter , typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    _Bi_iter __first,
    _Bi_iter __last,
    const basic_regex< _Ch_type, _Rx_traits > & __re,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a range.

Parameters

<code>__first</code>	[IN] The start of the string to search.
<code>__last</code>	[IN] One-past-the-end of the string to search.
<code>__re</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2353 of file `regex.h`.

References `std::regex_search()`.

2.70.3.61 `regex_search()` [3/7]

```
template<typename _Ch_type , class _Alloc , class _Rx_traits >
bool std::regex_search (
    const _Ch_type * __s,
    match_results< const _Ch_type *, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

<code>__s</code>	[IN] A C-string to search for the regex.
<code>__m</code>	[OUT] The set of regex matches.
<code>__e</code>	[IN] The regex to search for in <code>s</code> .
<code>__f</code>	[IN] The search flags.

Return values

<code>true</code>	A match was found within the string.
-------------------	--------------------------------------

Return values

<i>false</i>	No match was found within the string, the content of m is undefined.
--------------	--

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2376 of file `regex.h`.

References `std::regex_search()`.

2.70.3.62 `regex_search()` [4/7]

```
template<typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    const _Ch_type * __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a C-string.

Parameters

<i>__s</i>	[IN] The C-string to search.
<i>__e</i>	[IN] The regular expression to search for.
<i>__f</i>	[IN] Search policy flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2395 of file `regex.h`.

References `std::regex_search()`.

2.70.3.63 `regex_search()` [5/7]

```
template<typename _Ch_traits , typename _String_allocator , typename _Ch_type , typename _Rx_traits >
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __flags = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

<code>__s</code>	[IN] The string to search.
<code>__e</code>	[IN] The regular expression to search for.
<code>__flags</code>	[IN] Search policy flags.

Return values

<code>true</code>	A match was found within the string.
<code>false</code>	No match was found within the string.

Exceptions

<code>an</code>	exception of type <code>regex_error</code> .
-----------------	--

Definition at line 2414 of file `regex.h`.

References `std::regex_search()`.

2.70.3.64 `regex_search()` [6/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_search (
    const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
    match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > & __m,
    const basic_regex< _Ch_type, _Rx_traits > & __e,
    regex_constants::match_flag_type __f = regex_constants::match_default ) [inline]
```

Searches for a regular expression within a string.

Parameters

<code>__s</code>	[IN] A C++ string to search for the regex.
------------------	--

Parameters

<code>_↵ _m</code>	[OUT] The set of regex matches.
<code>_↵ _e</code>	[IN] The regex to search for in <i>s</i> .
<code>_↵ _f</code>	[IN] The search flags.

Return values

<i>true</i>	A match was found within the string.
<i>false</i>	No match was found within the string, the content of <i>m</i> is undefined.

Exceptions

<i>an</i>	exception of type <code>regex_error</code> .
-----------	--

Definition at line 2437 of file `regex.h`.

References `std::basic_string<_CharT, _Traits, _Alloc >::begin()`, `std::basic_string<_CharT, _Traits, _Alloc >::end()`, and `std::regex_search()`.

2.70.3.65 `regex_search()` [7/7]

```
template<typename _Ch_traits , typename _Ch_alloc , typename _Alloc , typename _Ch_type , typename
_Rx_traits >
bool std::regex_search (
    const basic_string<_Ch_type, _Ch_traits, _Ch_alloc > && ,
    match_results< typename basic_string<_Ch_type, _Ch_traits, _Ch_alloc >::const_↵
iterator, _Alloc > & ,
    const basic_regex<_Ch_type, _Rx_traits > & ,
    regex_constants::match_flag_type = regex_constants::match_default ) [delete]
```

Prevent unsafe attempts to get `match_results` from a temporary string.

2.70.3.66 `swap()` [1/2]

```
template<typename _Ch_type , typename _Rx_traits >
void swap (
    basic_regex<_Ch_type, _Rx_traits > & __lhs,
    basic_regex<_Ch_type, _Rx_traits > & __rhs ) [related]
```

Swaps the contents of two regular expression objects.

Parameters

<code>__lhs</code>	First regular expression.
<code>__rhs</code>	Second regular expression.

Definition at line 849 of file regex.h.

2.70.3.67 swap() [2/2]

```
template<typename _Bi_iter , typename _Alloc >
void std::swap (
    match_results< _Bi_iter, _Alloc > & __lhs,
    match_results< _Bi_iter, _Alloc > & __rhs ) [inline], [noexcept]
```

Swaps two match results.

Parameters

<code>__lhs</code>	A match result.
<code>__rhs</code>	A match result.

The contents of the two `match_results` objects are swapped.

Definition at line 2141 of file regex.h.

2.71 SGI

Collaboration diagram for SGI:



Classes

- class `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`
- struct `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`
- struct `__gnu_cxx::constant_unary_fun< _Result, _Argument >`
- struct `__gnu_cxx::constant_void_fun< _Result >`
- class `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >`
- class `__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >`
- struct `__gnu_cxx::project1st< _Arg1, _Arg2 >`
- struct `__gnu_cxx::project2nd< _Arg1, _Arg2 >`
- struct `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >`
- class `__gnu_cxx::rope< _CharT, _Alloc >`
- struct `__gnu_cxx::select1st< _Pair >`
- struct `__gnu_cxx::select2nd< _Pair >`
- class `__gnu_cxx::slist< _Tp, _Alloc >`
- class `__gnu_cxx::subtractive_rng`
- struct `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`
- class `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`

Functions

- `template<typename _Tp >`
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `size_t std::bitset< _Nb >::__Find_first () const noexcept`
- `size_t std::bitset< _Nb >::__Find_next (size_t __prev) const noexcept`
- `template<class _Operation1, class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`

- `template<class _Result >`
`constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Distance >`
`void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp __gnu_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`
`_RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`
`_OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`
- `bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos) noexcept`
- `bitset< _Nb > & std::bitset< _Nb >::Unchecked_set (size_t __pos, int __val) noexcept`
- `bitset< _Nb > & std::bitset< _Nb >::Unchecked_reset (size_t __pos) noexcept`
- `bitset< _Nb > & std::bitset< _Nb >::Unchecked_flip (size_t __pos) noexcept`
- `constexpr bool std::bitset< _Nb >::Unchecked_test (size_t __pos) const noexcept`

2.71.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the [online documentation](#), a copy of which is also shipped with the library source code (in `.../docs/html/documentation.html`). You can also read the documentation [on SGI's site](#), which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function `compose2` takes `f`, `g1`, and `g2`, and constructs the `binary_compose` instance for you. For example, if `f` returns an `int`, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.

As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.

`select1st` and `select2nd` are extensions provided by SGI. Their `operator()`s take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.

The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.

These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()`s completely ignore any arguments passed, and return the stored value.

- `constant_void_fun`'s `operator()` takes no arguments
- `constant_unary_fun`'s `operator()` takes one argument (ignored)
- `constant_binary_fun`'s `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

2.71.2 Function Documentation

2.71.2.1 `__median()` [1/2]

```
template<typename _Tp >
const _Tp& __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c )
```

Find the median of three values.

Parameters

\leftrightarrow _a	A value.
\leftrightarrow _b	A value.
\leftrightarrow _c	A value.

Returns

One of a, b or c.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $1 \leq m \leq n$ then the value returned will be m. This is an SGI extension.

Definition at line 539 of file ext/algorithm.

2.71.2.2 __median() [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp& __gnu_cxx::__median (
    const _Tp & __a,
    const _Tp & __b,
    const _Tp & __c,
    _Compare __comp )
```

Find the median of three values using a predicate for comparison.

Parameters

__a	A value.
__b	A value.
__c	A value.
__comp	A binary predicate.

Returns

One of a, b or c.

If $\{1, m, n\}$ is some convolution of $\{a, b, c\}$ such that $\text{comp}(1, m)$ and $\text{comp}(m, n)$ are both true then the value returned will be m. This is an SGI extension.

Definition at line 573 of file ext/algorithm.

2.71.2.3 `_Find_first()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_first ( ) const [inline], [noexcept]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or `size()` if not found.

See also

`_Find_next`

Definition at line 1373 of file `bitset`.

2.71.2.4 `_Find_next()`

```
template<size_t _Nb>
size_t std::bitset<_Nb>::_Find_next (
    size_t __prev ) const [inline], [noexcept]
```

Finds the index of the next "on" bit after `prev`.

Returns

The index of the next bit set, or `size()` if not found.

Parameters

<code>__prev</code>	Where to start searching.
---------------------	---------------------------

See also

`_Find_first`

Definition at line 1384 of file `bitset`.

2.71.2.5 `_Unchecked_flip()`

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_flip (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1058 of file `bitset`.

2.71.2.6 `_Unchecked_reset()`

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_reset (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1051 of file `bitset`.

2.71.2.7 `_Unchecked_set()` [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_set (
    size_t __pos ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1034 of file `bitset`.

2.71.2.8 `_Unchecked_set()` [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::_Unchecked_set (
    size_t __pos,
    int __val ) [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1041 of file `bitset`.

2.71.2.9 `_Unchecked_test()`

```
template<size_t _Nb>
constexpr bool std::bitset<_Nb>::_Unchecked_test (
    size_t __pos ) const [inline], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

Definition at line 1065 of file `bitset`.

2.71.2.10 compose1()

```
template<class _Operation1 , class _Operation2 >
binary_compose<_Operation1, _Operation2> __gnu_cxx::compose1 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2 ) [inline]
```

An [SGI extension](#) .

Definition at line 137 of file ext/functional.

2.71.2.11 compose2()

```
template<class _Operation1 , class _Operation2 , class _Operation3 >
binary_compose<_Operation1, _Operation2, _Operation3> __gnu_cxx::compose2 (
    const _Operation1 & __fn1,
    const _Operation2 & __fn2,
    const _Operation3 & __fn3 ) [inline]
```

An [SGI extension](#) .

Definition at line 164 of file ext/functional.

2.71.2.12 constant0()

```
template<class _Result >
constant_void_fun<_Result> __gnu_cxx::constant0 (
    const _Result & __val ) [inline]
```

An [SGI extension](#) .

Definition at line 322 of file ext/functional.

2.71.2.13 constant1()

```
template<class _Result >
constant_unary_fun<_Result, _Result> __gnu_cxx::constant1 (
    const _Result & __val ) [inline]
```

An [SGI extension](#) .

Definition at line 328 of file ext/functional.

2.71.2.14 `constant2()`

```
template<class _Result >
constant_binary_fun<_Result,_Result,_Result> __gnu_cxx::constant2 (
    const _Result & __val ) [inline]
```

An SGI extension .

Definition at line 334 of file ext/functional.

2.71.2.15 `copy_n()`

```
template<typename _InputIterator , typename _Size , typename _OutputIterator >
std::pair<_InputIterator, _OutputIterator> __gnu_cxx::copy_n (
    _InputIterator __first,
    _Size __count,
    _OutputIterator __result ) [inline]
```

Copies the range [first,first+count) into [result,result+count).

Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	The number of elements to copy.
<code>__result</code>	An output iterator.

Returns

A `std::pair` composed of `first+count` and `result+count`.

This is an SGI extension. This inline function will boil down to a call to `memmove` whenever possible. Failing that, if random access iterators are passed, then the loop count will be known (and therefore a candidate for compiler optimizations such as unrolling).

Definition at line 113 of file ext/algorithm.

2.71.2.16 `distance()`

```
template<typename _InputIterator , typename _Distance >
void __gnu_cxx::distance (
    _InputIterator __first,
    _InputIterator __last,
    _Distance & __n ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See <http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation+style.html>

Definition at line 105 of file ext/iterator.

2.71.2.17 identity_element() [1/2]

```
template<class _Tp >
_Tp __gnu_cxx::identity_element (
    std::plus< _Tp > ) [inline]
```

An [SGI extension](#) .

Definition at line 79 of file ext/functional.

2.71.2.18 identity_element() [2/2]

```
template<class _Tp >
_Tp __gnu_cxx::identity_element (
    std::multiplies< _Tp > ) [inline]
```

An [SGI extension](#) .

Definition at line 85 of file ext/functional.

2.71.2.19 lexicographical_compare_3way()

```
template<typename _InputIterator1 , typename _InputIterator2 >
int __gnu_cxx::lexicographical_compare_3way (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 )
```

memcmp on steroids.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

Definition at line 194 of file ext/algorithm.

2.71.2.20 power() [1/2]

```
template<typename _Tp , typename _Integer , typename _MonoidOperation >
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n,
    _MonoidOperation __monoid_op ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Definition at line 113 of file ext/numeric.

2.71.2.21 power() [2/2]

```
template<typename _Tp , typename _Integer >
_Tp __gnu_cxx::power (
    _Tp __x,
    _Integer __n ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Definition at line 123 of file ext/numeric.

2.71.2.22 random_sample() [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator >
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html

Definition at line 381 of file ext/algorithm.

2.71.2.23 `random_sample()` [2/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _RandomNumberGenerator >
_RandomAccessIterator __gnu_cxx::random_sample (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __out_first,
    _RandomAccessIterator __out_last,
    _RandomNumberGenerator & __rand ) [inline]
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html

Definition at line 404 of file ext/algorithm.

2.71.2.24 `random_sample_n()` [1/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Distance >
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n )
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html

Definition at line 260 of file ext/algorithm.

2.71.2.25 `random_sample_n()` [2/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Distance , typename _RandomNumberGenerator >
_OutputIterator __gnu_cxx::random_sample_n (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _OutputIterator __out,
    const _Distance __n,
    _RandomNumberGenerator & __rand )
```

This is an SGI extension.

Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_↵_style.html

Definition at line 294 of file ext/algorithm.

2.71.2.26 uninitialized_copy_n()

```
template<typename _InputIter , typename _Size , typename _ForwardIter >
std::pair<_InputIter, _ForwardIter> __gnu_cxx::uninitialized_copy_n (
    _InputIter __first,
    _Size __count,
    _ForwardIter __result ) [inline]
```

Copies the range [first,last) into result.

Parameters

<code>__first</code>	An input iterator.
<code>__count</code>	Length
<code>__result</code>	An output iterator.

Returns

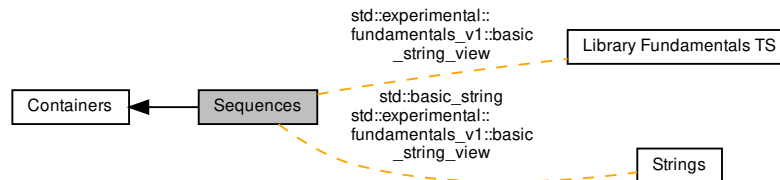
`__result + (__first + __count)`

Like `copy()`, but does not require an initialized output range.

Definition at line 121 of file `ext/memory`.

2.72 Sequences

Collaboration diagram for Sequences:



Classes

- struct `std::array<_Tp, _Nm>`
- class `std::basic_string<_CharT, _Traits, _Alloc>`
- class `std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits>`
- class `std::deque<_Tp, _Alloc>`
- class `std::forward_list<_Tp, _Alloc>`
- class `std::list<_Tp, _Alloc>`
- class `std::priority_queue<_Tp, _Sequence, _Compare>`
- class `std::queue<_Tp, _Sequence>`
- class `std::stack<_Tp, _Sequence>`
- class `std::vector<_Tp, _Alloc>`
- class `std::vector<bool, _Alloc>`

2.72.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in [tables](#).

2.73 Set Operations

Collaboration diagram for Set Operations:



Functions

- `template<typename _InputIterator1, typename _InputIterator2 >`
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

2.73.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

2.73.2 Function Documentation

2.73.2.1 `includes()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 >
constexpr bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2 ) [inline]
```

Determines whether all elements of a sequence exists in a range.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.

Returns

True if each element in `[__first2,__last2)` is contained in order within `[__first1,__last1)`. False otherwise.

This operation expects both `[__first1,__last1)` and `[__first2,__last2)` to be sorted. Searches for the presence of each element in `[__first2,__last2)` within `[__first1,__last1)`. The iterators over each range only move forward, so this is a linear algorithm. If an element in `[__first2,__last2)` is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2845 of file `stl_algo.h`.

2.73.2.2 `includes()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _Compare >
constexpr bool std::includes (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _Compare __comp ) [inline]
```

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

<code>__first1</code>	Start of search range.
<code>__last1</code>	End of search range.
<code>__first2</code>	Start of sequence
<code>__last2</code>	End of sequence.
<code>__comp</code>	Comparison function to use.

Returns

True if each element in [`__first2`,`__last2`) is contained in order within [`__first1`,`__last1`) according to `comp`. False otherwise.

This operation expects both [`__first1`,`__last1`) and [`__first2`,`__last2`) to be sorted. Searches for the presence of each element in [`__first2`,`__last2`) within [`__first1`,`__last1`), using `comp` to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [`__first2`,`__last2`) is not found before the search iterator reaches `__last2`, false is returned.

Definition at line 2891 of file `stl_algo.h`.

2.73.2.3 set_difference() [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
constexpr _OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5420 of file `stl_algo.h`.

2.73.2.4 set_difference() [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
constexpr _OutputIterator std::set_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to `__comp`, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5472 of file `stl_algo.h`.

2.73.2.5 set_intersection() [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
constexpr _OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the intersection of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5295 of file `stl_algo.h`.

2.73.2.6 set_intersection() [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
constexpr _OutputIterator std::set_intersection (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the intersection of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that iterator advances. If an element is contained in both ranges according to `__comp`, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5345 of file `stl_algo.h`.

2.73.2.7 `set_symmetric_difference()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
constexpr _OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the symmetric difference of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5553 of file `stl_algo.h`.

2.73.2.8 `set_symmetric_difference()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
constexpr _OutputIterator std::set_symmetric_difference (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
```

```
_InputIterator2 __last2,  
_OutputIterator __result,  
_Compare __comp ) [inline]
```

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5605 of file `stl_algo.h`.

2.73.2.9 `set_union()` [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
constexpr _OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Return the union of two sorted ranges.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the

iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5171 of file `stl_algo.h`.

2.73.2.10 `set_union()` [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
constexpr _OutputIterator std::set_union (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result,
    _Compare __comp ) [inline]
```

Return the union of two sorted ranges using a comparison functor.

Parameters

<code>__first1</code>	Start of first range.
<code>__last1</code>	End of first range.
<code>__first2</code>	Start of second range.
<code>__last2</code>	End of second range.
<code>__result</code>	Start of output range.
<code>__comp</code>	The comparison functor.

Returns

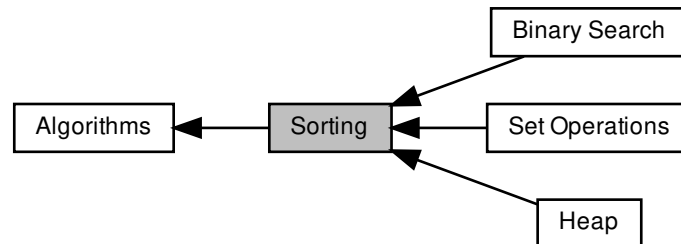
End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

Definition at line 5222 of file `stl_algo.h`.

2.74 Sorting

Collaboration diagram for Sorting:



Modules

- [Binary Search](#)
- [Heap](#)
- [Set Operations](#)

Functions

- `template<typename _BidirectionalIterator >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _II1, typename _II2 >`
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)`
- `template<typename _Tp >`
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`

- `template<typename _Tp, typename _Compare >`
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _ForwardIterator >`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`

- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`
`constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`
`constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`
`constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`
`void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

2.74.1 Detailed Description

2.74.2 Function Documentation

2.74.2.1 `inplace_merge()` [1/2]

```
template<typename _BidirectionalIterator >
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last ) [inline]
```

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Merges two sorted and consecutive ranges, [`__first`,`__middle`) and [`__middle`,`__last`), and puts the result in [`__first`,`__last`). The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(\text{__last} - \text{__first}) - 1$ comparisons. Otherwise an $N \log N$ algorithm is used, where N is `distance(__first, __last)`.

Definition at line 2590 of file `stl_algo.h`.

2.74.2.2 `inplace_merge()` [2/2]

```
template<typename _BidirectionalIterator, typename _Compare >
void std::inplace_merge (
    _BidirectionalIterator __first,
    _BidirectionalIterator __middle,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Merges two sorted ranges in place.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A functor to use for comparisons.

Returns

Nothing.

Merges two sorted and consecutive ranges, `[__first,__middle)` and `[middle,last)`, and puts the result in `[__first,__last)`. The output will be sorted. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

If enough additional memory is available, this takes $(\text{__last} - \text{__first}) - 1$ comparisons. Otherwise an $N \log N$ algorithm is used, where N is `distance(__first,__last)`.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 2631 of file `stl_algo.h`.

2.74.2.3 `is_sorted()` [1/2]

```
template<typename _ForwardIterator >
constexpr bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Determines whether the elements of a sequence are sorted.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3228 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.74.2.4 `is_sorted()` [2/2]

```
template<typename _ForwardIterator, typename _Compare>
constexpr bool std::is_sorted (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Determines whether the elements of a sequence are sorted according to a comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

True if the elements are sorted, false otherwise.

Definition at line 3243 of file `stl_algo.h`.

References `std::is_sorted_until()`.

2.74.2.5 `is_sorted_until()` [1/2]

```
template<typename _ForwardIterator>
constexpr _ForwardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Determines the end of a sorted sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 3274 of file `stl_algo.h`.

2.74.2.6 `is_sorted_until()` [2/2]

```
template<typename _ForwardIterator, typename _Compare >
constexpr _ForwardIterator std::is_sorted_until (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Determines the end of a sorted sequence using comparison functor.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator pointing to the last iterator `i` in `[__first, __last)` for which the range `[__first, i)` is sorted.

Definition at line 3299 of file `stl_algo.h`.

Referenced by `std::is_sorted()`.

2.74.2.7 `lexicographical_compare()` [1/2]

```
template<typename _II1, typename _II2 >
constexpr bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2 ) [inline]
```

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.

Returns

A boolean true or false.

Returns true if the sequence of elements defined by the range `[first1,last1)` is lexicographically less than the sequence of elements defined by the range `[first2,last2)`. Returns false otherwise. (Quoted from [25.3.8]/1.) If the iterators are all character pointers, then this is an inline call to `memcmp`.

Definition at line 1590 of file `stl_algobase.h`.

2.74.2.8 `lexicographical_compare()` [2/2]

```
template<typename _II1 , typename _II2 , typename _Compare >
constexpr bool std::lexicographical_compare (
    _II1 __first1,
    _II1 __last1,
    _II2 __first2,
    _II2 __last2,
    _Compare __comp ) [inline]
```

Performs **dictionary** comparison on ranges.

Parameters

<code>__first1</code>	An input iterator.
<code>__last1</code>	An input iterator.
<code>__first2</code>	An input iterator.
<code>__last2</code>	An input iterator.
<code>__comp</code>	A comparison functor .

Returns

A boolean true or false.

The same as the four-parameter `lexicographical_compare`, but uses the `comp` parameter instead of `<`.

Definition at line 1627 of file `stl_algobase.h`.

Referenced by `std::operator<()`.

2.74.2.9 `max()` [1/2]

```
template<typename _Tp >
constexpr const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The greater of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 254 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_nth_element()`, `std::_Deque_base<_StateSeqT, std::allocator<_StateSeqT> >::_M_initialize_map()`, `std::deque<_StateSeqT>::_M_reallocate_map()`, `std::discard_block_engine<__RandomNumberEngine, __p, __r>::max()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::max()`, `__gnu_parallel::multiseq_selection()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`.

2.74.2.10 `max()` [2/2]

```
template<typename _Tp , typename _Compare >
constexpr const _Tp & std::max (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The greater of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 300 of file `stl_algobase.h`.

2.74.2.11 `max_element()` [1/2]

```
template<typename _ForwardIterator >
constexpr _ForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return the maximum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the largest value.

Definition at line 5722 of file `stl_algo.h`.

2.74.2.12 `max_element()` [2/2]

```
template<typename _ForwardIterator , typename _Compare >
constexpr _ForwardIterator std::max_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return the maximum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the largest value according to `__comp`.

Definition at line 5747 of file `stl_algo.h`.

2.74.2.13 merge() [1/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator >
constexpr _OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
    _InputIterator2 __first2,
    _InputIterator2 __last2,
    _OutputIterator __result ) [inline]
```

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.

Returns

An output iterator equal to `__result + (__last1 - __first1)`
 • `(__last2 - __first2)`.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

Definition at line 4951 of file `stl_algo.h`.

2.74.2.14 merge() [2/2]

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
constexpr _OutputIterator std::merge (
    _InputIterator1 __first1,
    _InputIterator1 __last1,
```

```
_InputIterator2 __first2,  
_InputIterator2 __last2,  
_OutputIterator __result,  
_Compare __comp ) [inline]
```

Merges two sorted ranges.

Parameters

<code>__first1</code>	An iterator.
<code>__first2</code>	Another iterator.
<code>__last1</code>	Another iterator.
<code>__last2</code>	Another iterator.
<code>__result</code>	An iterator pointing to the end of the merged range.
<code>__comp</code>	A functor to use for comparisons.

Returns

An output iterator equal to `__result + (__last1 - __first1)`
 • `(__last2 - __first2)`.

Merges the ranges `[__first1, __last1)` and `[__first2, __last2)` into the sorted range `[__result, __result + (__last1 - __first1) + (__last2 - __first2))`. Both input ranges must be sorted, and the output range must not overlap with either of the input ranges. The sort is *stable*, that is, for equivalent elements in the two ranges, elements from the first range will always come before elements from the second.

The comparison function should have the same effects on ordering as the function used for the initial sort.

Definition at line 5002 of file `stl_algo.h`.

2.74.2.15 `min()` [1/2]

```
template<typename _Tp >
constexpr const _Tp & std::min (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

The lesser of the parameters.

This is the simple classic generic implementation. It will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 230 of file `stl_algobase.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, `__gnu_parallel::__parallel_sort_qs_divide()`, `std::__sample()`, `__gnu_parallel::__search_template()`, `__gnu_parallel::__sequential_random_shuffle()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `std::basic_string<char>::compare()`, `std::generate_canonical()`, `std::discard_block_engine<_RandomNumberEngine, __p, __r>::min()`, `std::shuffle_order_engine<_RandomNumberEngine, __k>::min()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_streambuf<_Elem, _Tr>::xsgetn()`, and `std::basic_streambuf<_Elem, _Tr>::xsputn()`.

2.74.2.16 `min()` [2/2]

```
template<typename _Tp, typename _Compare>
constexpr const _Tp & std::min (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

This does what you think it does.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

The lesser of the parameters.

This will work on temporary expressions, since they are only evaluated once, unlike a preprocessor macro.

Definition at line 278 of file `stl_algobase.h`.

2.74.2.17 `min_element()` [1/2]

```
template<typename _ForwardIterator>
constexpr _ForwardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return the minimum element in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

Iterator referencing the first instance of the smallest value.

Definition at line 5658 of file `stl_algo.h`.

2.74.2.18 min_element() [2/2]

```
template<typename _ForwardIterator, typename _Compare>
constexpr _ForwardIterator std::min_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return the minimum element in a range using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

Iterator referencing the first instance of the smallest value according to `__comp`.

Definition at line 5683 of file `stl_algo.h`.

2.74.2.19 minmax() [1/2]

```
template<typename _Tp>
constexpr pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b ) [inline]
```

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.

Returns

A pair(__b, __a) if __b is smaller than __a, pair(__a, __b) otherwise.

Definition at line 3325 of file stl_algo.h.

2.74.2.20 minmax() [2/2]

```
template<typename _Tp , typename _Compare >
constexpr pair< const _Tp &, const _Tp & > std::minmax (
    const _Tp & __a,
    const _Tp & __b,
    _Compare __comp ) [inline]
```

Determines min and max at once as an ordered pair.

Parameters

<code>__a</code>	A thing of arbitrary type.
<code>__b</code>	Another thing of arbitrary type.
<code>__comp</code>	A comparison functor .

Returns

A pair(__b, __a) if __b is smaller than __a, pair(__a, __b) otherwise.

Definition at line 3346 of file stl_algo.h.

2.74.2.21 minmax_element() [1/2]

```
template<typename _ForwardIterator >
constexpr pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last ) [inline]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

make_pair(m, M), where m is the first iterator i in [__first, __last) such that no other element in the range is smaller, and where M is the last iterator i in [__first, __last) such that no other element in the range is larger.

Definition at line 3426 of file stl_algo.h.

2.74.2.22 minmax_element() [2/2]

```
template<typename _ForwardIterator, typename _Compare >
constexpr pair<_ForwardIterator, _ForwardIterator> std::minmax_element (
    _ForwardIterator __first,
    _ForwardIterator __last,
    _Compare __comp ) [inline]
```

Return a pair of iterators pointing to the minimum and maximum elements in a range.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	Comparison functor.

Returns

make_pair(m, M), where m is the first iterator i in [__first, __last) such that no other element in the range is smaller, and where M is the last iterator i in [__first, __last) such that no other element in the range is larger.

Definition at line 3454 of file stl_algo.h.

2.74.2.23 next_permutation() [1/2]

```
template<typename _BidirectionalIterator >
constexpr bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Permute range into the next *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 2976 of file `stl_algo.h`.

2.74.2.24 next_permutation() [2/2]

```
template<typename _BidirectionalIterator , typename _Compare >
constexpr bool std::next_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Permute range into the next *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

Treats all permutations of the range [`__first`,`__last`) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the next one of this set. Returns true if there are more sequences to generate. If the sequence is the largest of the set, the smallest is generated and false returned.

Definition at line 3009 of file `stl_algo.h`.

2.74.2.25 nth_element() [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last ) [inline]
```

Sort a sequence just enough to find a particular position.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `*j < *i` is false.

Definition at line 4778 of file `stl_algo.h`.

2.74.2.26 nth_element() [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::nth_element (
    _RandomAccessIterator __first,
    _RandomAccessIterator __nth,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort a sequence just enough to find a particular position using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__nth</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Rearranges the elements in the range `[__first,__last)` so that `*__nth` is the same element that would have been in that position had the whole sequence been sorted. The elements either side of `*__nth` are not completely sorted, but for any iterator *i* in the range `[__first,__nth)` and any iterator *j* in the range `[__nth,__last)` it holds that `__comp(*j,*i)` is false.

Definition at line 4818 of file `stl_algo.h`.

2.74.2.27 `partial_sort()` [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last ) [inline]
```

Sort the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[first,last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*j < *i` and `*k < *i` are both false.

Definition at line 4702 of file `stl_algo.h`.

2.74.2.28 `partial_sort()` [2/2]

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::partial_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __middle,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__middle</code>	Another iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the smallest (`__middle-__first`) elements in the range `[__first,__last)` and moves them to the range `[__first,__middle)`. The order of the remaining elements in the range `[__middle,__last)` is undefined. After the sort if *i* and *j* are iterators in the range `[__first,__middle)` such that *i* precedes *j* and *k* is an iterator in the range `[__middle,__last)` then `*__comp(j,*i)` and `__comp(*k,*i)` are both false.

Definition at line 4741 of file `stl_algo.h`.

2.74.2.29 `partial_sort_copy()` [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator >
constexpr _RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
    _RandomAccessIterator __result_first,
    _RandomAccessIterator __result_last ) [inline]
```

Copy the smallest elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest *N* values from the range `[__first,__last)` to the range beginning at `__result_first`, where the number of elements to be copied, *N*, is the smaller of `(__last-__first)` and `(__result_last-__result_first)`. After the sort if *i* and *j* are iterators in the range `[__result_first,__result_first+N)` such that *i* precedes *j* then `*j<*i` is false. The value returned is `__result_first+N`.

Definition at line 1738 of file `stl_algo.h`.

2.74.2.30 `partial_sort_copy()` [2/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _Compare >
constexpr _RandomAccessIterator std::partial_sort_copy (
    _InputIterator __first,
    _InputIterator __last,
```

```
_RandomAccessIterator __result_first,  
_RandomAccessIterator __result_last,  
_Compare __comp ) [inline]
```

Copy the smallest elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An input iterator.
<code>__last</code>	Another input iterator.
<code>__result_first</code>	A random-access iterator.
<code>__result_last</code>	Another random-access iterator.
<code>__comp</code>	A comparison functor.

Returns

An iterator indicating the end of the resulting sequence.

Copies and sorts the smallest N values from the range $[\text{__first}, \text{__last})$ to the range beginning at `result_first`, where the number of elements to be copied, N , is the smaller of $(\text{__last} - \text{__first})$ and $(\text{__result_last} - \text{__result_first})$. After the sort if i and j are iterators in the range $[\text{__result_first}, \text{__result_first} + N)$ such that i precedes j then `__comp(*j, *i)` is false. The value returned is `__result_first + N`.

Definition at line 1789 of file `stl_algo.h`.

2.74.2.31 prev_permutation() [1/2]

```
template<typename _BidirectionalIterator >
constexpr bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last ) [inline]
```

Permute range into the previous *dictionary* ordering.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range as a set of *dictionary* sorted sequences. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3079 of file `stl_algo.h`.

2.74.2.32 prev_permutation() [2/2]

```
template<typename _BidirectionalIterator , typename _Compare >
constexpr bool std::prev_permutation (
    _BidirectionalIterator __first,
    _BidirectionalIterator __last,
    _Compare __comp ) [inline]
```

Permute range into the previous *dictionary* ordering using comparison functor.

Parameters

<code>__first</code>	Start of range.
<code>__last</code>	End of range.
<code>__comp</code>	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

Treats all permutations of the range [`__first`,`__last`) as a set of *dictionary* sorted sequences ordered by `__comp`. Permutes the current sequence into the previous one of this set. Returns true if there are more sequences to generate. If the sequence is the smallest of the set, the largest is generated and false returned.

Definition at line 3112 of file stl_algo.h.

2.74.2.33 `sort()` [1/2]

```
template<typename _RandomAccessIterator >
constexpr void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort the elements of a sequence.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator *i* in the range `[__first,__last-1)`,

$*(i+1) < *i$ is false.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4856 of file `stl_algo.h`.

2.74.2.34 `sort()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare >
constexpr void std::sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

Sorts the elements in the range `[__first, __last)` in ascending order, such that `__comp(*(i+1), *i)` is false for every iterator `i` in the range `[__first, __last-1)`.

The relative ordering of equivalent elements is not preserved, use `stable_sort()` if this is needed.

Definition at line 4887 of file `stl_algo.h`.

2.74.2.35 `stable_sort()` [1/2]

```
template<typename _RandomAccessIterator >
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last ) [inline]
```

Sort the elements of a sequence, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.

Returns

Nothing.

Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator `i` in the range `[__first,__last-1)`, `*(i+1)<*(i)` is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `x<y` is false and `y<x` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5065 of file `stl_algo.h`.

2.74.2.36 `stable_sort()` [2/2]

```
template<typename _RandomAccessIterator, typename _Compare>
void std::stable_sort (
    _RandomAccessIterator __first,
    _RandomAccessIterator __last,
    _Compare __comp ) [inline]
```

Sort the elements of a sequence using a predicate for comparison, preserving the relative order of equivalent elements.

Parameters

<code>__first</code>	An iterator.
<code>__last</code>	Another iterator.
<code>__comp</code>	A comparison functor.

Returns

Nothing.

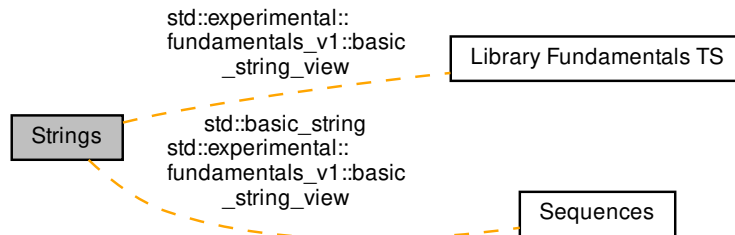
Sorts the elements in the range `[__first,__last)` in ascending order, such that for each iterator `i` in the range `[__first,__last-1)`, `__comp(*(i+1),*(i))` is false.

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first,__last)` such that `__comp(x,y)` is false and `__comp(y,x)` is false will have the same relative ordering after calling `stable_sort()`.

Definition at line 5099 of file `stl_algo.h`.

2.75 Strings

Collaboration diagram for Strings:



Classes

- class `std::basic_string< _CharT, _Traits, _Alloc >`
- class `std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >`
- struct `std::char_traits< _CharT >`

Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

2.75.1 Detailed Description

2.75.2 Typedef Documentation

2.75.2.1 string

```
typedef basic_string<char> std::string
```

A string of `char`.

Definition at line 74 of file `stringfwd.h`.

2.75.2.2 u16string

```
typedef basic_string<char16_t> std::u16string
```

A string of `char16_t`.

Definition at line 93 of file `stringfwd.h`.

2.75.2.3 u32string

```
typedef basic_string<char32_t> std::u32string
```

A string of `char32_t`.

Definition at line 96 of file `stringfwd.h`.

2.75.2.4 wstring

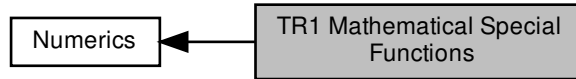
```
typedef basic_string<wchar_t> std::wstring
```

A string of `wchar_t`.

Definition at line 83 of file `stringfwd.h`.

2.76 TR1 Mathematical Special Functions

Collaboration diagram for TR1 Mathematical Special Functions:



Functions

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int __n, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_laguerref` (unsigned int __n, unsigned int __m, float __x)
- `long double std::tr1::assoc_laguerrel` (unsigned int __n, unsigned int __m, long double __x)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int __l, unsigned int __m, _Tp __x)
- `float std::tr1::assoc_legendref` (unsigned int __l, unsigned int __m, float __x)
- `long double std::tr1::assoc_legendrel` (unsigned int __l, unsigned int __m, long double __x)
- `template<typename _Tpx, typename _Tpy >`
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (_Tpx __x, _Tpy __y)
- `float std::tr1::betaf` (float __x, float __y)
- `long double std::tr1::betal` (long double __x, long double __y)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (_Tp __k)
- `float std::tr1::comp_ellint_1f` (float __k)
- `long double std::tr1::comp_ellint_1l` (long double __k)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (_Tp __k)
- `float std::tr1::comp_ellint_2f` (float __k)
- `long double std::tr1::comp_ellint_2l` (long double __k)
- `template<typename _Tp, typename _Tpn >`
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (_Tp __k, _Tpn __nu)
- `float std::tr1::comp_ellint_3f` (float __k, float __nu)
- `long double std::tr1::comp_ellint_3l` (long double __k, long double __nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg` (_Tpa __a, _Tpc __c, _Tp __x)
- `float std::tr1::conf_hypergf` (float __a, float __c, float __x)
- `long double std::tr1::conf_hypergl` (long double __a, long double __c, long double __x)
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (_Tpnu __nu, _Tp __x)
- `float std::tr1::cyl_bessel_if` (float __nu, float __x)
- `long double std::tr1::cyl_bessel_il` (long double __nu, long double __x)
- `template<typename _Tpnu, typename _Tp >`
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j` (_Tpnu __nu, _Tp __x)

- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_bessel_k** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp >
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type **std::tr1::cyl_neumann** (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp >
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type **std::tr1::ellint_1** (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp >
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type **std::tr1::ellint_2** (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp >
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type **std::tr1::ellint_3** (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::expint** (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::hermite** (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type **std::tr1::hyperg** (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float **std::tr1::hypergf** (float __a, float __b, float __c, float __x)
- long double **std::tr1::hypergl** (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::laguerre** (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::legendre** (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::riemann_zeta** (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp >
__gnu_cxx::__promote< _Tp >::__type **std::tr1::sph_bessel** (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)

- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre` (unsigned int __l, unsigned int __m, _Tp __theta)
- `float std::tr1::sph_legendref` (unsigned int __l, unsigned int __m, float __theta)
- `long double std::tr1::sph_legendrel` (unsigned int __l, unsigned int __m, long double __theta)
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann` (unsigned int __n, _Tp __x)
- `float std::tr1::sph_neumannf` (unsigned int __n, float __x)
- `long double std::tr1::sph_neumannl` (unsigned int __n, long double __x)

2.76.1 Detailed Description

A collection of advanced mathematical special functions.

2.76.2 Function Documentation

2.76.2.1 `assoc_laguerre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_laguerre (
    unsigned int __n,
    unsigned int __m,
    _Tp __x ) [inline]
```

5.2.1.1 Associated Laguerre polynomials.

Definition at line 1275 of file tr1/cmath.

2.76.2.2 `assoc_legendre()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::assoc_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __x ) [inline]
```

5.2.1.2 Associated Legendre functions.

Definition at line 1292 of file tr1/cmath.

2.76.2.3 beta()

```
template<typename _Tpx , typename _Tpy >
__gnu_cxx::__promote_2<_Tpx, _Tpy>::__type std::tr1::beta (
    _Tpx __x,
    _Tpy __y ) [inline]
```

5.2.1.3 Beta functions.

Definition at line 1309 of file tr1/cmath.

2.76.2.4 comp_ellint_1()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_1 (
    _Tp __k ) [inline]
```

5.2.1.4 Complete elliptic integrals of the first kind.

Definition at line 1326 of file tr1/cmath.

2.76.2.5 comp_ellint_2()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::comp_ellint_2 (
    _Tp __k ) [inline]
```

5.2.1.5 Complete elliptic integrals of the second kind.

Definition at line 1343 of file tr1/cmath.

2.76.2.6 comp_ellint_3()

```
template<typename _Tp , typename _Tpn >
__gnu_cxx::__promote_2<_Tp, _Tpn>::__type std::tr1::comp_ellint_3 (
    _Tp __k,
    _Tpn __nu ) [inline]
```

5.2.1.6 Complete elliptic integrals of the third kind.

Definition at line 1360 of file tr1/cmath.

2.76.2.7 `conf_hyperg()`

```
template<typename _Tpa , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_3<_Tpa, _Tpc, _Tp>::__type std::tr1::conf_hyperg (
    _Tpa __a,
    _Tpc __c,
    _Tp __x ) [inline]
```

5.2.1.7 Confluent hypergeometric functions.

Definition at line 1678 of file tr1/cmath.

2.76.2.8 `cyl_bessel_i()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_i (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

5.2.1.8 Regular modified cylindrical Bessel functions.

Definition at line 1377 of file tr1/cmath.

2.76.2.9 `cyl_bessel_j()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_j (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

5.2.1.9 Cylindrical Bessel functions (of the first kind).

Definition at line 1394 of file tr1/cmath.

2.76.2.10 `cyl_bessel_k()`

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_bessel_k (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

5.2.1.10 Irregular modified cylindrical Bessel functions.

Definition at line 1411 of file tr1/cmath.

2.76.2.11 cyl_neumann()

```
template<typename _Tpnu , typename _Tp >
__gnu_cxx::__promote_2<_Tpnu, _Tp>::__type std::tr1::cyl_neumann (
    _Tpnu __nu,
    _Tp __x ) [inline]
```

5.2.1.11 Cylindrical Neumann functions.

Definition at line 1428 of file tr1/cmath.

2.76.2.12 ellint_1()

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_1 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

5.2.1.12 Incomplete elliptic integrals of the first kind.

Definition at line 1445 of file tr1/cmath.

2.76.2.13 ellint_2()

```
template<typename _Tp , typename _Tpp >
__gnu_cxx::__promote_2<_Tp, _Tpp>::__type std::tr1::ellint_2 (
    _Tp __k,
    _Tpp __phi ) [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

Definition at line 1462 of file tr1/cmath.

2.76.2.14 ellint_3()

```
template<typename _Tp , typename _Tpn , typename _Tpp >
__gnu_cxx::__promote_3<_Tp, _Tpn, _Tpp>::__type std::tr1::ellint_3 (
    _Tp __k,
    _Tpn __nu,
    _Tpp __phi ) [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

Definition at line 1479 of file tr1/cmath.

2.76.2.15 expint()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::expint (
    _Tp __x ) [inline]
```

5.2.1.15 Exponential integrals.

Definition at line 1496 of file tr1/cmath.

2.76.2.16 hermite()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::hermite (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.16 Hermite polynomials.

Definition at line 1513 of file tr1/cmath.

2.76.2.17 hyperg()

```
template<typename _Tpa , typename _Tpb , typename _Tpc , typename _Tp >
__gnu_cxx::__promote_4<_Tpa, _Tpb, _Tpc, _Tp>::__type std::tr1::hyperg (
    _Tpa __a,
    _Tpb __b,
    _Tpc __c,
    _Tp __x ) [inline]
```

5.2.1.17 Hypergeometric functions.

Definition at line 1695 of file tr1/cmath.

2.76.2.18 laguerre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::laguerre (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.18 Laguerre polynomials.

Definition at line 1530 of file tr1/cmath.

2.76.2.19 legendre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::legendre (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.19 Legendre polynomials.

Definition at line 1547 of file tr1/cmath.

2.76.2.20 riemann_zeta()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::riemann_zeta (
    _Tp __x ) [inline]
```

5.2.1.20 Riemann zeta function.

Definition at line 1564 of file tr1/cmath.

2.76.2.21 sph_bessel()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_bessel (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.21 Spherical Bessel functions.

Definition at line 1581 of file tr1/cmath.

2.76.2.22 sph_legendre()

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_legendre (
    unsigned int __l,
    unsigned int __m,
    _Tp __theta ) [inline]
```

5.2.1.22 Spherical associated Legendre functions.

Definition at line 1598 of file tr1/cmath.

2.76.2.23 sph_neumann()

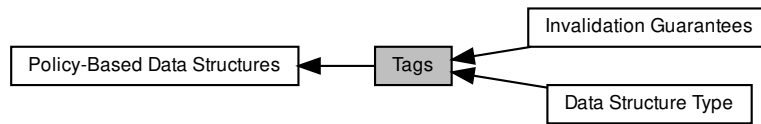
```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::tr1::sph_neumann (
    unsigned int __n,
    _Tp __x ) [inline]
```

5.2.1.23 Spherical Neumann functions.

Definition at line 1615 of file tr1/cmath.

2.77 Tags

Collaboration diagram for Tags:



Modules

- [Data Structure Type](#)
- [Invalidation Guarantees](#)

Classes

- [struct `__gnu_pbds::trivial_iterator_tag`](#)

Typedefs

- [typedef void `__gnu_pbds::trivial_iterator_difference_type`](#)

2.77.1 Detailed Description

2.77.2 Typedef Documentation

2.77.2.1 `trivial_iterator_difference_type`

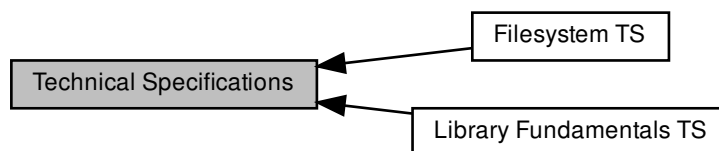
```
typedef void \_\_gnu\_pbds::trivial\_iterator\_difference\_type
```

Prohibit moving trivial iterators.

Definition at line 79 of file `tag_and_trait.hpp`.

2.78 Technical Specifications

Collaboration diagram for Technical Specifications:



Modules

- [Filesystem TS](#)
- [Library Fundamentals TS](#)

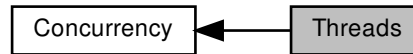
2.78.1 Detailed Description

Components specified by various Technical Specifications.

As indicated by the `std::experimental` namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.

2.79 Threads

Collaboration diagram for Threads:



Namespaces

- [std::this_thread](#)

Classes

- struct [std::hash< thread::id >](#)
- class [std::thread::id](#)
- class [std::thread](#)

Functions

- bool **std::operator!=** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- bool **std::operator<** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- template<class _CharT, class _Traits >
[basic_ostream](#)< _CharT, _Traits > & **std::operator<<** ([basic_ostream](#)< _CharT, _Traits > &__out, [thread::id](#) __id)
- bool **std::operator<=** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- bool **std::operator==** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- bool **std::operator>** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- bool **std::operator>=** ([thread::id](#) __x, [thread::id](#) __y) noexcept
- void **std::swap** ([thread](#) &__x, [thread](#) &__y) noexcept

2.79.1 Detailed Description

Classes for thread support.

2.80 Time

Collaboration diagram for Time:



Files

- file [chrono](#)

Namespaces

- [std::chrono](#)
- [std::literals::chrono_literals](#)

Classes

- struct [std::common_type< chrono::duration< _Rep, _Period > >](#)
- struct [std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >](#)
- struct [std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >](#)
- struct [std::common_type< chrono::time_point< _Clock, _Duration > >](#)
- struct [std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >](#)
- struct [std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >](#)
- struct [std::chrono::duration< _Rep, _Period >](#)
- struct [std::chrono::duration_values< _Rep >](#)
- struct [std::chrono::_V2::steady_clock](#)
- struct [std::chrono::_V2::system_clock](#)
- struct [std::chrono::time_point< _Clock, _Dur >](#)
- struct [std::chrono::treat_as_floating_point< _Rep >](#)

Macros

- `#define _GLIBCXX_CHRONO_INT64_T`

Typedefs

- using `std::chrono::_V2::high_resolution_clock` = `system_clock`
- using `std::chrono::hours` = `duration< int64_t, ratio< 3600 > >`
- using `std::chrono::microseconds` = `duration< int64_t, micro >`
- using `std::chrono::milliseconds` = `duration< int64_t, milli >`
- using `std::chrono::minutes` = `duration< int64_t, ratio< 60 > >`
- using `std::chrono::nanoseconds` = `duration< int64_t, nano >`
- using `std::chrono::seconds` = `duration< int64_t >`

Functions

- `template<typename _ToDur, typename _Rep, typename _Period >`
`constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur >`
`constexpr enable_if< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > >::type std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`
`constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period, typename _Rep2 >`
`constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, __Period > &__d, const _Rep2 &__s)`
- `template<typename _Rep1, typename _Rep2, typename _Period >`
`constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2 >`
`constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2 >`
`constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)`
- `template<typename _Clock, typename _Dur1, typename _Dur2 >`
`constexpr common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)`

2.80.1 Detailed Description

Classes and functions for time.

2.80.2 Typedef Documentation

2.80.2.1 high_resolution_clock

```
using std::chrono::_V2::high_resolution_clock = typedef system_clock
```

Highest-resolution clock.

This is the clock "with the shortest tick period." Alias to `std::system_clock` until higher-than-nanosecond definitions become feasible.

Definition at line 1100 of file `chrono`.

2.80.2.2 hours

```
using std::chrono::hours = typedef duration< int64_t , ratio<3600> >
```

hours

Definition at line 792 of file `chrono`.

2.80.2.3 microseconds

```
using std::chrono::microseconds = typedef duration< int64_t , micro>
```

microseconds

Definition at line 780 of file `chrono`.

2.80.2.4 milliseconds

```
using std::chrono::milliseconds = typedef duration< int64_t , milli>
```

milliseconds

Definition at line 783 of file `chrono`.

2.80.2.5 minutes

```
using std::chrono::minutes = typedef duration< int64_t , ratio< 60> >
```

minutes

Definition at line 789 of file chrono.

2.80.2.6 nanoseconds

```
using std::chrono::nanoseconds = typedef duration< int64_t , nano>
```

nanoseconds

Definition at line 777 of file chrono.

2.80.2.7 seconds

```
using std::chrono::seconds = typedef duration< int64_t >
```

seconds

Definition at line 786 of file chrono.

2.80.3 Function Documentation

2.80.3.1 duration_cast()

```
template<typename _ToDur , typename _Rep , typename _Period >  
constexpr __enable_if_is_duration<_ToDur> std::chrono::duration_cast (   
    const duration< _Rep, _Period > & __d )
```

duration_cast

Definition at line 254 of file chrono.

2.80.3.2 operator*() [1/2]

```
template<typename _Rep1 , typename _Period , typename _Rep2 >
constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (
    const duration< _Rep1, _Period > & __d,
    const _Rep2 & __s ) [related]
```

Multiply a duration by a scalar value.

Definition at line 637 of file chrono.

2.80.3.3 operator*() [2/2]

```
template<typename _Rep1 , typename _Rep2 , typename _Period >
constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > operator* (
    const _Rep1 & __s,
    const duration< _Rep2, _Period > & __d ) [related]
```

Multiply a duration by a scalar value.

Definition at line 647 of file chrono.

2.80.3.4 operator+() [1/3]

```
template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+
(
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs ) [related]
```

The sum of two durations.

Definition at line 594 of file chrono.

2.80.3.5 operator+() [2/3]

```
template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >
operator+ (
    const time_point< _Clock, _Dur1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs ) [related]
```

Adjust a time point forwards by the given duration.

Definition at line 917 of file chrono.

2.80.3.6 operator+() [3/3]

```
template<typename _Rep1 , typename _Period1 , typename _Clock , typename _Dur2 >
constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type >
operator+ (
    const duration< _Rep1, _Period1 > & __lhs,
    const time_point< _Clock, _Dur2 > & __rhs ) [related]
```

Adjust a time point forwards by the given duration.

Definition at line 931 of file chrono.

2.80.3.7 operator-() [1/3]

```
template<typename _Rep1 , typename _Period1 , typename _Rep2 , typename _Period2 >
constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator-
(
    const duration< _Rep1, _Period1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs ) [related]
```

The difference between two durations.

Definition at line 608 of file chrono.

2.80.3.8 operator-() [2/3]

```
template<typename _Clock , typename _Dur1 , typename _Rep2 , typename _Period2 >
constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type >
operator- (
    const time_point< _Clock, _Dur1 > & __lhs,
    const duration< _Rep2, _Period2 > & __rhs ) [related]
```

Adjust a time point backwards by the given duration.

Definition at line 945 of file chrono.

2.80.3.9 operator-() [3/3]

```
template<typename _Clock , typename _Dur1 , typename _Dur2 >
constexpr common_type< _Dur1, _Dur2 >::type operator- (
    const time_point< _Clock, _Dur1 > & __lhs,
    const time_point< _Clock, _Dur2 > & __rhs ) [related]
```

The difference between two time points (as a duration)

Definition at line 961 of file chrono.

2.80.3.10 time_point_cast()

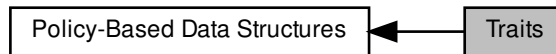
```
template<typename _ToDur , typename _Clock , typename _Dur >
constexpr enable_if<__is_duration<_ToDur>::value, time_point<_Clock, _ToDur> >::type std::chrono<
::time_point_cast (
    const time_point< _Clock, _Dur > & __t )
```

time_point_cast

Definition at line 873 of file chrono.

2.81 Traits

Collaboration diagram for Traits:



Classes

- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >`
- struct `__gnu_pbds::container_traits< Cntr >`
- struct `__gnu_pbds::container_traits_base< _Tag >`
- struct `__gnu_pbds::container_traits_base< binary_heap_tag >`
- struct `__gnu_pbds::container_traits_base< binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< cc_hash_tag >`
- struct `__gnu_pbds::container_traits_base< gp_hash_tag >`
- struct `__gnu_pbds::container_traits_base< list_update_tag >`
- struct `__gnu_pbds::container_traits_base< ov_tree_tag >`
- struct `__gnu_pbds::container_traits_base< pairing_heap_tag >`
- struct `__gnu_pbds::container_traits_base< pat_trie_tag >`
- struct `__gnu_pbds::container_traits_base< rb_tree_tag >`
- struct `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >`
- struct `__gnu_pbds::container_traits_base< splay_tree_tag >`
- struct `__gnu_pbds::container_traits_base< thin_heap_tag >`
- struct `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, Mapped >`
- struct `__gnu_pbds::detail::no_throw_copies< Key, null_type >`
- struct `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`
- struct `__gnu_pbds::null_type`
- struct `__gnu_pbds::detail::rebind_traits< _Alloc, T >`
- struct `__gnu_pbds::detail::select_value_type< Key, Mapped >`
- struct `__gnu_pbds::detail::select_value_type< Key, null_type >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >`
- struct `__gnu_pbds::detail::stored_data< _Tv, _Th, false >`
- struct `__gnu_pbds::detail::stored_hash< _Th >`
- struct `__gnu_pbds::detail::stored_value< _Tv >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`

- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >`
- struct `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >`
- struct `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >`
- struct `__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >`

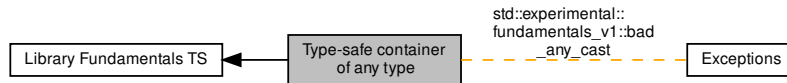
Variables

- static `null_type __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type`

2.81.1 Detailed Description

2.82 Type-safe container of any type

Collaboration diagram for Type-safe container of any type:



Classes

- class `std::experimental::fundamentals_v1::any`
- class `std::experimental::fundamentals_v1::bad_any_cast`

Macros

- `#define __cpp_lib_experimental_any`

Functions

- static void `std::experimental::fundamentals_v1::any::Manager_internal<_Tp>::_S_manage` (`_Op __op` which, const `any *__anyp`, `_Arg *__arg`)
- static void `std::experimental::fundamentals_v1::any::Manager_external<_Tp>::_S_manage` (`_Op __op` which, const `any *__anyp`, `_Arg *__arg`)
- template<typename `_ValueType`>
`_ValueType std::experimental::fundamentals_v1::any_cast` (const `any &__any`)
- void `std::experimental::fundamentals_v1::swap` (`any &__x`, `any &__y`) noexcept
- template<typename `_ValueType`>
`_ValueType std::experimental::fundamentals_v1::any_cast` (`any &__any`)
- template<typename `_ValueType` , typename enable_if<is_move_constructible< `_ValueType` >::value||is_lvalue_reference< `_ValueType` >::value, bool >::type = true>
`_ValueType std::experimental::fundamentals_v1::any_cast` (`any &&__any`)
- template<typename `_ValueType`>
const `_ValueType *` `std::experimental::fundamentals_v1::any_cast` (const `any *__any`) noexcept
- template<typename `_ValueType`>
`_ValueType *` `std::experimental::fundamentals_v1::any_cast` (`any *__any`) noexcept

2.82.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

2.82.2 Function Documentation

2.82.2.1 `any_cast()` [1/5]

```
template<typename _ValueType >
_ValueType std::experimental::fundamentals_v1::any_cast (
    const any & __any ) [inline]
```

Access the contained object.

Template Parameters

<code>_ValueType</code>	A const-reference or CopyConstructible type.
-------------------------	--

Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

Returns

The contained object.

Exceptions

<code>bad_any_cast</code>	If <code>__any.type() != typeid(remove_reference_t<_ValueType>)</code>
---------------------------	--

Definition at line 355 of file `experimental/any`.

2.82.2.2 `any_cast()` [2/5]

```
template<typename _ValueType >
_ValueType std::experimental::fundamentals_v1::any_cast (
    any & __any ) [inline]
```

Access the contained object.

Template Parameters

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--

Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

Returns

The contained object.

Exceptions

<i>bad_any_cast</i>	If <code>__any.type() != typeid(remove_reference_t<_ValueType>)</code>
---------------------	--

Definition at line 378 of file experimental/any.

2.82.2.3 `any_cast()` [3/5]

```
template<typename _ValueType , typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
    any && __any ) [inline]
```

Access the contained object.

Template Parameters

<code>_ValueType</code>	A reference or CopyConstructible type.
-------------------------	--

Parameters

<code>__any</code>	The object to access.
--------------------	-----------------------

Returns

The contained object.

Exceptions

<i>bad_any_cast</i>	If <code>__any.type() != typeid(remove_reference_t<_ValueType>)</code>
---------------------	--

Definition at line 392 of file experimental/any.

2.82.2.4 `any_cast()` [4/5]

```
template<typename _ValueType >
const _ValueType* std::experimental::fundamentals_v1::any_cast (
    const any * __any ) [inline], [noexcept]
```

Access the contained object.

Template Parameters

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

Parameters

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

Definition at line 468 of file `experimental/any`.

2.82.2.5 `any_cast()` [5/5]

```
template<typename _ValueType >
_ValueType* std::experimental::fundamentals_v1::any_cast (
    any * __any ) [inline], [noexcept]
```

Access the contained object.

Template Parameters

<code>_ValueType</code>	The type of the contained object.
-------------------------	-----------------------------------

Parameters

<code>__any</code>	A pointer to the object to access.
--------------------	------------------------------------

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

Definition at line 476 of file `experimental/any`.

2.82.2.6 swap()

```
void std::experimental::fundamentals_v1::swap (
    any & __x,
    any & __y ) [inline], [noexcept]
```

Exchange the states of two `any` objects.

Definition at line 342 of file `experimental/any`.

2.83 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Classes

- struct `std::uniform_real_distribution<_RealType>::param_type`
- class `std::uniform_real_distribution<_RealType>`

Functions

- template<typename _IntType >
bool `std::operator!=` (const `std::uniform_int_distribution<_IntType>` &__d1, const `std::uniform_int_distribution<_IntType>` &__d2)
- template<typename _IntType >
bool `std::operator!=` (const `std::uniform_real_distribution<_IntType>` &__d1, const `std::uniform_real_distribution<_IntType>` &__d2)
- template<typename _IntType, typename _CharT, typename _Traits >
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::uniform_int_distribution<_IntType>` &)
- template<typename _RealType, typename _CharT, typename _Traits >
`std::basic_ostream<_CharT, _Traits>` & `std::operator<<` (`std::basic_ostream<_CharT, _Traits>` &, const `std::uniform_real_distribution<_RealType>` &)
- template<typename _IntType, typename _CharT, typename _Traits >
`std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::uniform_int_distribution<_IntType>` &)
- template<typename _RealType, typename _CharT, typename _Traits >
`std::basic_istream<_CharT, _Traits>` & `std::operator>>` (`std::basic_istream<_CharT, _Traits>` &, `std::uniform_real_distribution<_RealType>` &)

2.83.1 Detailed Description

2.83.2 Function Documentation

2.83.2.1 operator!=() [1/2]

```
template<typename _IntType >
bool std::operator!=(
    const std::uniform_int_distribution< _IntType > & __d1,
    const std::uniform_int_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform integer distributions have different parameters.

Definition at line 1698 of file random.h.

2.83.2.2 operator!=() [2/2]

```
template<typename _IntType >
bool std::operator!=(
    const std::uniform_real_distribution< _IntType > & __d1,
    const std::uniform_real_distribution< _IntType > & __d2 ) [inline]
```

Return true if two uniform real distributions have different parameters.

Definition at line 1919 of file random.h.

2.83.2.3 operator<<() [1/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_int_distribution< _IntType > & __x )
```

Inserts a uniform_int_distribution random number distribution __x into the output stream os.

Parameters

<code>__os</code>	An output stream.
<code>__x</code>	A uniform_int_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 867 of file bits/random.tcc.

References `std::basic_ios< _CharT, _Traits >::fill()`, `std::ios_base::flags()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

2.83.2.4 operator<<() [2/2]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_ostream< _CharT, _Traits > & std::operator<< (
    std::basic_ostream< _CharT, _Traits > & __os,
    const std::uniform_real_distribution< _RealType > & __x )
```

Inserts a uniform_real_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A uniform_real_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

Definition at line 925 of file bits/random.tcc.

References std::basic_ios< _CharT, _Traits >::fill(), std::ios_base::flags(), std::ios_base::precision(), and std::basic_istream< _CharT, _Traits >::widen().

2.83.2.5 operator>>() [1/2]

```
template<typename _IntType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::uniform_int_distribution< _IntType > & __x )
```

Extracts a uniform_int_distribution random number distribution __x from the input stream __is.

Parameters

__is	An input stream.
__x	A uniform_int_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

Definition at line 887 of file bits/random.tcc.

References std::dec(), std::ios_base::flags(), std::uniform_int_distribution< _IntType >::param(), and std::skipws().

2.83.2.6 operator>>() [2/2]

```
template<typename _RealType , typename _CharT , typename _Traits >
std::basic_istream< _CharT, _Traits > & std::operator>> (
    std::basic_istream< _CharT, _Traits > & __is,
    std::uniform_real_distribution< _RealType > & __x )
```

Extracts a uniform_real_distribution random number distribution __x from the input stream __is.

Parameters

\leftrightarrow __is	An input stream.
\leftrightarrow __x	A uniform_real_distribution random number generator engine.

Returns

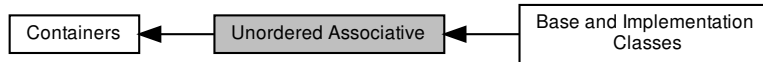
The input stream with __x extracted or in an error state.

Definition at line 948 of file bits/random.tcc.

References std::ios_base::flags(), std::uniform_real_distribution< _RealType >::param(), and std::skipws().

2.84 Unordered Associative

Collaboration diagram for Unordered Associative:



Modules

- [Base and Implementation Classes](#)

Classes

- class `std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::unordered_set<_Value, _Hash, _Pred, _Alloc>`

2.84.1 Detailed Description

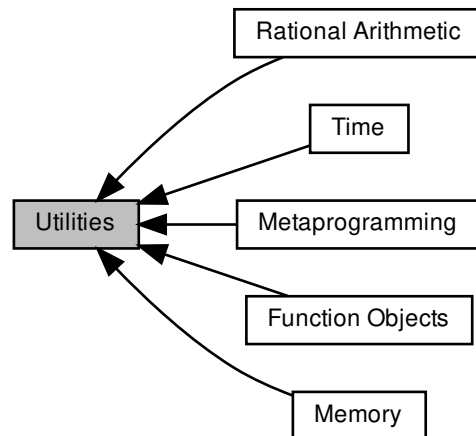
Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in [tables](#).

2.85 Utilities

Collaboration diagram for Utilities:



Modules

- [Function Objects](#)
- [Memory](#)
- [Metaprogramming](#)
- [Rational Arithmetic](#)
- [Time](#)

Classes

- `struct std::_Tuple_impl< _Idx, _Elements >`
- `struct std::_Tuple_impl< _Idx, _Head, _Tail... >`
- `class std::bitset< _Nb >`
- `struct std::pair< _T1, _T2 >`
- `struct std::piecewise_construct_t`
- `class std::bitset< _Nb >::reference`
- `class std::tuple< _Elements >`
- `class std::tuple< _T1, _T2 >`
- `struct std::tuple_element< 0, tuple< _Head, _Tail... > >`
- `struct std::tuple_element< __i, tuple< _Head, _Tail... > >`
- `struct std::tuple_element< __i, tuple< > >`
- `struct std::tuple_size< tuple< _Elements... > >`
- `struct std::type_index`
- `struct std::uses_allocator< tuple< _Types... >, _Alloc >`

Macros

- `#define __cpp_lib_tuples_by_type`

Typedefs

- `template<typename _Res, typename _Callable, typename... _Args>`
`using std::__can_invoke_as_nonvoid = __enable_if_t<__and<__not<is_void<_Res>>, is_convertible<`
`typename __invoke_result<_Callable, _Args...>::type, _Res>>::value, _Res>`
- `template<typename _Res, typename _Callable, typename... _Args>`
`using std::__can_invoke_as_void = __enable_if_t<__and<is_void<_Res>, __is_invocable<_Callable,`
`_Args...>>::value, _Res>`
- `template<typename _Tp>`
`using std::__empty_not_final = typename conditional<__is_final(_Tp), false_type, __empty_non_tuple<_Tp>>::type`

Functions

- `template<typename... _Args1, typename... _Args2>`
`constexpr std::pair<_T1, _T2>::pair (piecewise_construct_t, tuple<_Args1...>, tuple<_Args2...>)`
- `template<typename _Tp>`
`constexpr _Tp * std::__addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`
`constexpr _Tp std::__exchange (_Tp &__obj, _Up &&__new_val)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`
`constexpr _Head & std::__get_helper (_Tuple_impl<__i, _Head, _Tail...> &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`
`constexpr const _Head & std::__get_helper (const _Tuple_impl<__i, _Head, _Tail...> &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`
`constexpr _Head & std::__get_helper2 (_Tuple_impl<__i, _Head, _Tail...> &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`
`constexpr const _Head & std::__get_helper2 (const _Tuple_impl<__i, _Head, _Tail...> &__t) noexcept`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`
`constexpr _Up && std::__invfwd (typename remove_reference<_Tp>::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>`
`constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (_Callable &&__fn, _Args &&... __args)`
`noexcept(__is_nothrow_invocable<_Callable, _Args...>::value)`
- `template<typename _Res, typename _Fn, typename... _Args>`
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`
`constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`
`constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
`args)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`
`constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp>`
`constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _Callable, typename... _Args>`
`constexpr __can_invoke_as_nonvoid<_Res, _Callable, _Args...> std::__invoke_r (_Callable &&__fn, _Args`
`&&... __args)`

- `template<typename _Tp >`
`constexpr _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp >`
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`
`constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept`
- `template<typename... _Elements>`
`constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept`
- `template<std::size_t __i, typename... _Elements>`
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`
`constexpr _Tp & std::get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`
`constexpr _Tp && std::get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`
`constexpr const _Tp & std::get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`
`constexpr const _Tp && std::get (const tuple< _Types... > &&__t) noexcept`
- `template<typename _T1, typename _T2 >`
`constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > > make_pair (_T1 &&__x, _T2 &&__y)`
- `template<typename... _Elements>`
`constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)`
- `template<typename _Tp >`
`constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move_if_noexcept (_Tp &__x) noexcept`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`
`constexpr bool std::operator>= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`

- `template<typename _Tp >`
`constexpr enable_if< __and< __not< __is_tuple_like< _Tp > , is_move_constructible< _Tp > ,`
`is_move_assignable< _Tp > >::value >::type std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */)`
`is_nothrow_move_assignable< _Tp >>`
- `template<typename _Tp , size_t _Nm>`
`constexpr enable_if< __is_swappable< _Tp >::value >::type std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm])`
`noexcept(/*conditional */)`
- `template<typename... _Elements>`
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _↵`
`Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`
`constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`
`constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type`

Variables

- `constexpr _Swallow_assign std::ignore`
- `constexpr piecewise_construct_t std::piecewise_construct`

- `template<typename _T1 , typename _T2 >`
`constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr bool operator< (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr bool operator!= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr bool operator> (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr bool operator<= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr bool operator>= (const pair< _T1, _T2 > &__x, const pair< _T1, _T2 > &__y)`
- `template<typename _T1 , typename _T2 >`
`constexpr enable_if< __and< __is_swappable< _T1 > , __is_swappable< _T2 > >::value >::type swap (pair<`
`_T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))`

2.85.1 Detailed Description

Components deemed generally useful. Includes pair, tuple, forward/move helpers, ratio, function object, metaprogramming and type traits, time, date, and memory functions.

2.85.2 Function Documentation

2.85.2.1 pair()

```
template<class _T1, class _T2>
template<typename... _Args1, typename... _Args2>
constexpr std::pair<_T1, _T2>::pair (
    piecewise_construct_t ,
    tuple<_Args1...> __first,
    tuple<_Args2...> __second ) [inline]
```

"piecewise construction" using a tuple of arguments for each member.

Parameters

<code>__first</code>	Arguments for the first member of the pair.
<code>__second</code>	Arguments for the second member of the pair.

The elements of each tuple will be used as the constructor arguments for the data members of the pair.

Definition at line 1678 of file tuple.

2.85.2.2 `__addressof()`

```
template<typename _Tp >
constexpr _Tp* std::__addressof (
    _Tp & __r ) [inline], [noexcept]
```

Same as C++11 `std::addressof`.

Definition at line 49 of file move.h.

Referenced by `std::_Destroy()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `std::addressof()`, `std::list< __inp, __rebind_inp >::merge()`, `std::sub_match< _Bi_iter >::operator<()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::list< __inp, __rebind_inp >::operator=()`, `std::sub_match< _Bi_iter >::operator==()`, `std::forward_list< _Tp, _Alloc >::remove()`, `std::list< __inp, __rebind_inp >::remove()`, `std::rethrow_if_nested()`, and `std::list< __inp, __rebind_inp >::splice()`.

2.85.2.3 `__invoke()`

```
template<typename _Callable , typename... _Args>
constexpr __invoke_result<_Callable, _Args...>::type std::__invoke (
    _Callable && __fn,
    _Args &&... __args ) [noexcept]
```

Invoke a callable object.

Definition at line 89 of file invoke.h.

2.85.2.4 `addressof()`

```
template<typename _Tp >
constexpr _Tp* std::addressof (
    _Tp & __r ) [inline], [noexcept]
```

Returns the actual address of the object or function referenced by `r`, even in the presence of an overloaded operator`&`.

Parameters

↩	Reference to an object or function.
↩	
↩	
↩	
<i>r</i>	

Returns

The actual address.

Definition at line 140 of file move.h.

References `std::__addressof()`.

Referenced by `std::pointer_traits<_Tp*>::pointer_to()`.

2.85.2.5 forward() [1/2]

```
template<typename _Tp >
constexpr _Tp&& std::forward (
    typename std::remove_reference< _Tp >::type & __t ) [noexcept]
```

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 76 of file move.h.

2.85.2.6 forward() [2/2]

```
template<typename _Tp >
constexpr _Tp&& std::forward (
    typename std::remove_reference< _Tp >::type && __t ) [noexcept]
```

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Definition at line 87 of file move.h.

2.85.2.7 forward_as_tuple()

```
template<typename... _Elements>
constexpr tuple<_Elements&&...> std::forward_as_tuple (
    _Elements &&... __args ) [noexcept]
```

std::forward_as_tuple

Definition at line 1490 of file tuple.

2.85.2.8 get() [1/8]

```
template<std::size_t __i, typename... _Elements>
constexpr __tuple_element_t<__i, tuple<_Elements...> >& std::get (
    tuple< _Elements... > & __t ) [noexcept]
```

Return a reference to the ith element of a tuple.

Definition at line 1298 of file tuple.

2.85.2.9 get() [2/8]

```
template<std::size_t __i, typename... _Elements>
constexpr const __tuple_element_t<__i, tuple<_Elements...> >& std::get (
    const tuple< _Elements... > & __t ) [noexcept]
```

Return a const reference to the ith element of a const tuple.

Definition at line 1304 of file tuple.

2.85.2.10 get() [3/8]

```
template<std::size_t __i, typename... _Elements>
constexpr __tuple_element_t<__i, tuple<_Elements...> >&& std::get (
    tuple< _Elements... > && __t ) [noexcept]
```

Return an rvalue reference to the ith element of a tuple rvalue.

Definition at line 1310 of file tuple.

2.85.2.11 `get()` [4/8]

```
template<std::size_t __i, typename... _Elements>
constexpr const __tuple_element_t<__i, tuple<_Elements...> >&& std::get (
    const tuple<_Elements...> && __t ) [noexcept]
```

Return a const rvalue reference to the *i*th element of a const tuple rvalue.

Definition at line 1319 of file tuple.

2.85.2.12 `get()` [5/8]

```
template<typename _Tp, typename... _Types>
constexpr _Tp& std::get (
    tuple<_Types...> & __t ) [noexcept]
```

Return a reference to the unique element of type *_Tp* of a tuple.

Definition at line 1342 of file tuple.

2.85.2.13 `get()` [6/8]

```
template<typename _Tp, typename... _Types>
constexpr _Tp&& std::get (
    tuple<_Types...> && __t ) [noexcept]
```

Return a reference to the unique element of type *_Tp* of a tuple rvalue.

Definition at line 1348 of file tuple.

2.85.2.14 `get()` [7/8]

```
template<typename _Tp, typename... _Types>
constexpr const _Tp& std::get (
    const tuple<_Types...> & __t ) [noexcept]
```

Return a const reference to the unique element of type *_Tp* of a tuple.

Definition at line 1354 of file tuple.

2.85.2.15 `get()` [8/8]

```
template<typename _Tp , typename... _Types>
constexpr const _Tp&& std::get (
    const tuple< _Types... > && __t ) [noexcept]
```

Return a const reference to the unique element of type `_Tp` of a const tuple rvalue.

Definition at line 1361 of file `tuple`.

2.85.2.16 `make_pair()`

```
template<typename _T1 , typename _T2 >
constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (
    _T1 && __x,
    _T2 && __y ) [related]
```

A convenience wrapper for creating a pair from two objects.

Parameters

<code>__x</code>	The first object.
<code>__y</code>	The second object.

Returns

A newly-constructed `pair<>` object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).

Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.

Definition at line 567 of file `std_pair.h`.

2.85.2.17 `move()`

```
template<typename _Tp >
constexpr std::remove_reference<_Tp>::type&& std::move (
    _Tp && __t ) [noexcept]
```

Convert a value to an rvalue.

Parameters

↔	A thing of arbitrary type.
↔	
↔	
↔	
<i>t</i>	

Returns

The parameter cast to an rvalue-reference to allow moving it.

Definition at line 101 of file move.h.

Referenced by `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, `std::shared_ptr< _State >::atomic_compare_exchange_strong_explicit()`, `std::function< _Res(_ArgTypes...)>::function()`, `std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert()`, `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`, `std::unordered_map< _Key, _Tp, _Hash, _Pred >::insert()`, `std::map< _Key, _Tp, _Compare, _Alloc >::insert()`, `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`, `std::list< __inp, __rebind_inp >::insert()`, `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert()`, `std::deque< _StateSeqT >::insert()`, `std::forward_list< _Tp, _Alloc >::insert_after()`, `std::forward_list< _Tp, _Alloc >::merge()`, `std::move_if_noexcept()`, `std::function< _Res(_ArgTypes...)>::operator=()`, `std::basic_regex< _Ch_type, _Rx_traits >::operator=()`, `std::forward_list< _Tp, _Alloc >::operator=()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::operator=()`, `std::list< __inp, __rebind_inp >::operator=()`, `std::deque< _StateSeqT >::operator=()`, `std::unordered_map< _Key, _Tp, _Hash, _Pred >::operator[]()`, `std::forward_list< _Tp, _Alloc >::push_front()`, `std::unique_ptr< _Result< _Res > >::reset()`, `std::unique_ptr< _Tp[], _Dp >::reset()`, `std::list< __inp, __rebind_inp >::splice()`, and `std::unique_ptr< _Result< _Res > >::~~unique_ptr()`.

2.85.2.18 move_if_noexcept()

```
template<typename _Tp >
constexpr conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp&, _Tp&&>::type std::move_if_noexcept (
    _Tp & __x ) [noexcept]
```

Conditionally convert a value to an rvalue.

Parameters

↔	A thing of arbitrary type.
<i>__x</i>	

Returns

The parameter, possibly cast to an rvalue-reference.

Same as `std::move` unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Definition at line 121 of file move.h.

References `std::move()`.

2.85.2.19 `operator!=()`

```
template<typename _T1 , typename _T2 >
constexpr bool operator!= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Uses `operator==` to find the result.

Definition at line 496 of file stl_pair.h.

2.85.2.20 `operator<()`

```
template<typename _T1 , typename _T2 >
constexpr bool operator< (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Defines a lexicographical order for pairs.

For two pairs of the same type, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

Definition at line 489 of file stl_pair.h.

2.85.2.21 `operator<=()`

```
template<typename _T1 , typename _T2 >
constexpr bool operator<= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Uses `operator<` to find the result.

Definition at line 508 of file stl_pair.h.

2.85.2.22 operator==()

```
template<typename _T1 , typename _T2 >
constexpr bool operator== (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Two pairs of the same type are equal iff their members are equal.

Definition at line 466 of file stl_pair.h.

2.85.2.23 operator>()

```
template<typename _T1 , typename _T2 >
constexpr bool operator> (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Uses operator< to find the result.

Definition at line 502 of file stl_pair.h.

2.85.2.24 operator>=()

```
template<typename _T1 , typename _T2 >
constexpr bool operator>= (
    const pair< _T1, _T2 > & __x,
    const pair< _T1, _T2 > & __y ) [related]
```

Uses operator< to find the result.

Definition at line 514 of file stl_pair.h.

2.85.2.25 swap() [1/4]

```
template<typename _Tp >
constexpr enable_if<__and<__not<__is_tuple_like<_Tp> >, is_move_constructible<_Tp>, is_move_assignable<_Tp> >::value>::type std::swap (
    _Tp & __a,
    _Tp & __b ) [inline], [noexcept]
```

Swaps two values.

Parameters

\leftrightarrow _a	A thing of arbitrary type.
\leftrightarrow _b	Another thing of arbitrary type.

Returns

Nothing.

Definition at line 189 of file move.h.

2.85.2.26 swap() [2/4]

```
template<typename _Tp , size_t _Nm>
constexpr enable\_if<__is_swappable<_Tp>::value>::type std::swap (
    _Tp(&) __a[_Nm],
    _Tp(&) __b[_Nm] ) [inline], [noexcept]
```

Swap the contents of two arrays.

Definition at line 213 of file move.h.

2.85.2.27 swap() [3/4]

```
template<typename _T1 , typename _T2 >
constexpr enable\_if<__and< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap
(
    pair< _T1, _T2 > & __x,
    pair< _T1, _T2 > & __y ) [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

Definition at line 533 of file stl_pair.h.

2.85.2.28 swap() [4/4]

```
template<typename... _Elements>
constexpr enable_if<__and<__is_swappable<_Elements>...>::value >::type std::swap (
    tuple< _Elements... > & __x,
    tuple< _Elements... > & __y ) [inline], [delete], [noexcept]
```

swap

Definition at line 1633 of file tuple.

2.85.2.29 tie()

```
template<typename... _Elements>
constexpr tuple<_Elements&...> std::tie (
    _Elements &... __args ) [noexcept]
```

tie

Definition at line 1619 of file tuple.

Referenced by std::basic_ios< char, _Traits >::copyfmt().

2.85.2.30 tuple_cat()

```
template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>↵
::value>::type>
constexpr auto std::tuple_cat (
    _Tpls &&... __tpls ) -> typename __tuple_cat_result<_Tpls...>::__type
```

tuple_cat

Definition at line 1605 of file tuple.

2.85.3 Variable Documentation**2.85.3.1 piecewise_construct**

```
constexpr piecewise_construct_t std::piecewise_construct [inline]
```

Tag for piecewise construction of std::pair objects.

Definition at line 83 of file stl_pair.h.

3 Namespace Documentation

3.1 `__gnu_cxx` Namespace Reference

Namespaces

- [__detail](#)
- [typelist](#)

Classes

- [struct __alloc_traits](#)
- [struct __common_pool_policy](#)
- [class __mt_alloc](#)
- [class __mt_alloc_base](#)
- [struct __per_type_pool_policy](#)
- [class __pool](#)
- [class __pool< false >](#)
- [class __pool< true >](#)
- [class __pool_alloc](#)
- [class __pool_alloc_base](#)
- [struct __pool_base](#)
- [class __rc_string_base](#)
- [class __scoped_lock](#)
- [class __versa_string](#)
- [struct _Caster](#)
- [struct _Char_types](#)
- [class _ExtPtr_allocator](#)
- [struct _Invalid_type](#)
- [class _Pointer_adapter](#)
- [class _Relative_pointer_impl](#)
- [class _Relative_pointer_impl< const _Tp >](#)
- [class _Std_pointer_impl](#)
- [struct _Unqualified_type](#)
- [struct annotate_base](#)
- [class binary_compose](#)
- [class bitmap_allocator](#)
- [struct char_traits](#)
- [struct character](#)
- [struct condition_base](#)
- [struct constant_binary_fun](#)
- [struct constant_unary_fun](#)
- [struct constant_void_fun](#)
- [class debug_allocator](#)
- [class enc_filebuf](#)
- [struct encoding_char_traits](#)
- [class encoding_state](#)
- [struct forced_error](#)
- [class free_list](#)

- class [hash_map](#)
- class [hash_multimap](#)
- class [hash_multiset](#)
- class [hash_set](#)
- struct [limit_condition](#)
- class [malloc_allocator](#)
- class [new_allocator](#)
- struct [project1st](#)
- struct [project2nd](#)
- struct [random_condition](#)
- struct [rb_tree](#)
- class [recursive_init_error](#)
- class [rope](#)
- struct [select1st](#)
- struct [select2nd](#)
- class [slist](#)
- class [stdio_filebuf](#)
- class [stdio_sync_filebuf](#)
- class [subtractive_rng](#)
- struct [temporary_buffer](#)
- class [throw_allocator_base](#)
- struct [throw_allocator_limit](#)
- struct [throw_allocator_random](#)
- struct [throw_value_base](#)
- struct [throw_value_limit](#)
- struct [throw_value_random](#)
- class [unary_compose](#)

Typedefs

- typedef void(* **__destroy_handler**) (void *)
- template<typename `_Tp` >
using **__int_traits** = `__numeric_traits_integer<_Tp>`
- typedef `__versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base >` **__rc_string**
- typedef `__vstring` **__sso_string**
- typedef `__versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base >` **__u16rc_string**
- typedef `__u16vstring` **__u16sso_string**
- typedef `__versa_string< char16_t >` **__u16vstring**
- typedef `__versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base >` **__u32rc_string**
- typedef `__u32vstring` **__u32sso_string**
- typedef `__versa_string< char32_t >` **__u32vstring**
- typedef `__versa_string< char >` **__vstring**
- typedef `__versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base >` **__wrc_string**
- typedef `__wvstring` **__wsso_string**
- typedef `__versa_string< wchar_t >` **__wvstring**
- typedef `rope< char >` **crope**
- typedef `rope< wchar_t >` **wrope**

Enumerations

- enum { **_S_num_primes** }
- enum **_Lock_policy** { **_S_single**, **_S_mutex**, **_S_atomic** }

Functions

- void **__atomic_add** (volatile **_Atomic_word** *, int) noexcept
- void **__atomic_add_dispatch** (**_Atomic_word** * __mem, int __val)
- void **__atomic_add_single** (**_Atomic_word** * __mem, int __val)
- template<class **_Tp** >
void **__aux_require_boolean_expr** (const **_Tp** & __t)
- template<typename **_ToType**, typename **_FromType** >
_ToType **__const_pointer_cast** (const **_FromType** & __arg)
- template<typename **_ToType**, typename **_FromType** >
_ToType **__const_pointer_cast** (**_FromType** * __arg)
- template<typename **_InputIterator**, typename **_Size**, typename **_OutputIterator** >
std::pair< **_InputIterator**, **_OutputIterator** > **__copy_n** (**_InputIterator** __first, **_Size** __count, **_OutputIterator** __↔
__result, [std::input_iterator_tag](#))
- template<typename **_RAIterator**, typename **_Size**, typename **_OutputIterator** >
std::pair< **_RAIterator**, **_OutputIterator** > **__copy_n** (**_RAIterator** __first, **_Size** __count, **_OutputIterator** __result,
[std::random_access_iterator_tag](#))
- template<typename **_InputIterator**, typename **_Distance** >
void **__distance** (**_InputIterator** __first, **_InputIterator** __last, **_Distance** & __n, [std::input_iterator_tag](#))
- template<typename **_RandomAccessIterator**, typename **_Distance** >
void **__distance** (**_RandomAccessIterator** __first, **_RandomAccessIterator** __last, **_Distance** & __n, [std::random_access_iterator_tag](#))
- template<typename **_ToType**, typename **_FromType** >
_ToType **__dynamic_pointer_cast** (const **_FromType** & __arg)
- template<typename **_ToType**, typename **_FromType** >
_ToType **__dynamic_pointer_cast** (**_FromType** * __arg)
- void **__error_type_must_be_a_signed_integer_type** ()
- void **__error_type_must_be_an_integer_type** ()
- void **__error_type_must_be_an_unsigned_integer_type** ()
- **_Atomic_word** **__exchange_and_add** (volatile **_Atomic_word** *, int) noexcept
- **_Atomic_word** **__exchange_and_add_dispatch** (**_Atomic_word** * __mem, int __val)
- **_Atomic_word** **__exchange_and_add_single** (**_Atomic_word** * __mem, int __val)
- template<class **_Concept** >
constexpr void **__function_requires** ()
- template<typename **_Type** >
bool **__is_null_pointer** (**_Type** * __ptr)
- template<typename **_Type** >
bool **__is_null_pointer** (**_Type**)
- bool **__is_null_pointer** (std::nullptr_t)
- template<typename **_InputIterator1**, typename **_InputIterator2** >
int **__lexicographical_compare_3way** (**_InputIterator1** __first1, **_InputIterator1** __last1, **_InputIterator2** __first2,
_InputIterator2 __last2)
- int **__lexicographical_compare_3way** (const unsigned char * __first1, const unsigned char * __last1, const
unsigned char * __first2, const unsigned char * __last2)
- int **__lexicographical_compare_3way** (const char * __first1, const char * __last1, const char * __first2, const
char * __last2)

- `template<typename _Tp >`
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`
`const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `crope::reference __mutable_reference_at (crope &__c, std::size_t __i)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`
`_Tp __power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`
`_Tp __power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`
`_RandomAccessIterator __random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __reinterpret_pointer_cast (_FromType * __arg)`
- `_Slist_node_base * __slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __↵ node)`
- `_Slist_node_base * __slist_reverse (_Slist_node_base * __node)`
- `std::size_t __slist_size (_Slist_node_base * __node)`
- `void __slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __↵ __before_last)`
- `void __slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >`
`_ToType __static_pointer_cast (_FromType * __arg)`
- `size_t __stl_hash_string (const char * __s)`
- `unsigned long __stl_next_prime (unsigned long __n)`
- `template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>`
`_Ret __stoa (_TRet(* __convf)(const _CharT *, _CharT **, _Base...), const char * __name, const _CharT * __str, std::size_t * __idx, _Base... __base)`
- `void __throw_concurrency_lock_error ()`
- `void __throw_concurrency_unlock_error ()`
- `void __throw_forced_error ()`
- `template<typename _String, typename _CharT = typename _String::value_type>`
`_String __to_xstring (int(* __convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT * __fmt,...)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`
`std::pair< _RandomAccessIter, _ForwardIter > __uninitialized_copy_n (_RandomAccessIter __first, _Size __↵ __count, _ForwardIter __result, std::random_access_iterator_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)`

- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a (_InputIter __first, _Size __count, _ForwardIter`
`__result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`
`std::pair< _InputIter, _ForwardIter > __uninitialized_copy_n_a (_InputIter __first, _Size __count, _ForwardIter`
`__result, std::allocator< _Tp >)`
- `void __verbose_terminate_handler ()`
- `std::size_t _Bit_scan_forward (std::size_t __num)`
- `template<typename _ForwardIterator, typename _Allocator >`
`void __Destroy_const (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)`
- `template<typename _ForwardIterator, typename _Tp >`
`void __Destroy_const (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)`
- `template<class _CharT, class _Traits >`
`void __Rope_fill (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)`
- `template<class _CharT >`
`bool __Rope_is_simple (_CharT *)`
- `bool __Rope_is_simple (char *)`
- `bool __Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`
`void __Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT >`
`void __S_cond_store_eos (_CharT &)`
- `void __S_cond_store_eos (char &__c)`
- `void __S_cond_store_eos (wchar_t &__c)`
- `template<class _CharT >`
`_CharT __S_eos (_CharT *)`
- `template<class _CharT >`
`bool __S_is_basic_char_type (_CharT *)`
- `bool __S_is_basic_char_type (char *)`
- `bool __S_is_basic_char_type (wchar_t *)`
- `template<class _CharT >`
`bool __S_is_one_byte_char_type (_CharT *)`
- `bool __S_is_one_byte_char_type (char *)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type airy_ai (_Tp __x)`
- `float airy_aif (float __x)`
- `long double airy_ail (long double __x)`
- `template<typename _Tp >`
`__gnu_cxx::__promote< _Tp >::__type airy_bi (_Tp __x)`
- `float airy_bif (float __x)`
- `long double airy_bil (long double __x)`
- `template<class _Operation1, class _Operation2 >`
`unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`
`binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float conf_hypergf (float __a, float __c, float __x)`
- `long double conf_hypergl (long double __a, long double __c, long double __x)`

- `template<class _Result >`
`constant_void_fun< _Result > constant0 (const _Result &__val)`
- `template<class _Result >`
`constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)`
- `template<class _Result >`
`constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`
`std::pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __↵
result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`
`void count (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`
`void count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator, typename _Distance >`
`void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float hypergf (float __a, float __b, float __c, float __x)`
- `long double hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<class _Tp >`
`_Tp identity_element (std::plus< _Tp >)`
- `template<class _Tp >`
`_Tp identity_element (std::multiplies< _Tp >)`
- `template<typename _InputIterator1, typename _InputIterator2 >`
`int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵
_InputIterator2 __last2)`
- `template<class _Ret, class _Tp, class _Arg >`
`std::mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`
`std::const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun1 (_Ret(_Tp::* __f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg >`
`std::mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`
`std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun1_ref (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Tp >`
`bool operator!= (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value,
_HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator!= (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key,
_Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset<
_Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >`
`bool operator!= (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap<
_Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`

- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`
`bool operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`
`bool operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator!= (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)`
- `template<typename _Tp, typename _Poolp >`
`bool operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<class _Tp, class _Alloc >`
`bool operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<typename _Tp, typename _Cond >`
`bool operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`constexpr bool operator!= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Iterator, typename _Container >`
`constexpr bool operator!= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator!= (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`

- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Iterator, typename _Container >`
`constexpr __normal_iterator< _Iterator, _Container > operator+ (typename __normal_iterator< _Iterator, _Container >::difference_type __n, const __normal_iterator< _Iterator, _Container > &__i) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (___versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (___versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (___versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`__versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (___versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`

- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`
`rope< _CharT, _Alloc > & operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`
`_Rope_const_iterator< _CharT, _Alloc > operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`std::ptrdiff_t operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`_Rope_iterator< _CharT, _Alloc > operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`
`std::ptrdiff_t operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Cond >`
`throw_value_base< _Cond > operator- (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`constexpr auto operator- (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept -> decltype(__lhs.base() - __rhs.base())`
- `template<typename _Iterator, typename _Container >`
`constexpr __normal_iterator< _Iterator, _Container >::difference_type operator- (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _Value, typename _Int, typename _St >`
`bool operator< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Cond >`
`bool operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<class _Tp, class _Alloc >`
`bool operator< (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator< (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`constexpr bool operator< (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator< (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator< (const rope< _CharT, _Alloc > & __left, const rope< _CharT, _Alloc > & __right)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const ←
__gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __x)`
- `std::ostream & operator<< (std::ostream &os, const annotate_base & __b)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::beta_distribution< _RealType > & __x)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
_Pointer_adapter< _StoreT > & __p)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
rice_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
nakagami_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
pareto_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
k_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
arcsine_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
hoyt_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::triangular_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::von_mises_distribution< _RealType > & __x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::hypergeometric_distribution< _UIntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
logistic_distribution< _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const
__gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > & __x)`

- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`
`__gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<class _CharT, class _Traits, class _Alloc >`
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__o, const`
`rope< _CharT, _Alloc > &__r)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`
`bool operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`
`bool operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >`
`bool operator<= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`
`bool operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _↵`
`_IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`
`constexpr bool operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator<`
`_Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _↵`
`_CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`
`bool operator<= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT,`
`_Alloc > &__y)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &↵`
`__y)`
- `template<class _CharT, class _Alloc >`
`bool operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Value, typename _Int, typename _St >`
`bool operator== (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)`
- `template<typename _Tp >`
`bool operator== (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)`
- `template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All >`
`bool operator== (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key,`
`_HF, _Ex, _Eq, _All > &__ht2)`
- `template<class _Value, class _HashFcn, class _EqualKey, class _Alloc >`
`bool operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value,`
`_HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`
`bool operator== (const hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map< _Key,`
`_Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`

- ```
bool operator==(const __gnu_cxx::simd_fast_mersenne_twister_engine<_UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4> &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine<_UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4> &__rhs)
```
- `template<class _CharT, class _Alloc>`  
`bool operator==(const _Rope_char_ptr_proxy<_CharT, _Alloc> &__x, const _Rope_char_ptr_proxy<_CharT, _Alloc> &__y)`
  - `template<class _CharT, class _Alloc>`  
`bool operator==(const _Rope_const_iterator<_CharT, _Alloc> &__x, const _Rope_const_iterator<_CharT, _Alloc> &__y)`
  - `template<class _CharT, class _Alloc>`  
`bool operator==(const _Rope_iterator<_CharT, _Alloc> &__x, const _Rope_iterator<_CharT, _Alloc> &__y)`
  - `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>`  
`bool operator==(const hash_multiset<_Val, _HashFcn, _EqualKey, _Alloc> &__hs1, const hash_multiset<_Val, _HashFcn, _EqualKey, _Alloc> &__hs2)`
  - `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>`  
`bool operator==(const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm1, const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc> &__hm2)`
  - `template<typename _Tp1, typename _Tp2>`  
`bool operator==(const _Pointer_adapter<_Tp1> &__lhs, const _Pointer_adapter<_Tp2> &__rhs)`
  - `template<typename _Tp1, typename _Tp2>`  
`bool operator==(const _Pointer_adapter<_Tp1> &__lhs, _Tp2 __rhs)`
  - `template<typename _Tp1, typename _Tp2>`  
`bool operator==(const _Tp1 __lhs, const _Pointer_adapter<_Tp2> &__rhs)`
  - `template<typename _Tp>`  
`bool operator==(const _Pointer_adapter<_Tp> &__lhs, int __rhs)`
  - `template<typename _Tp>`  
`bool operator==(int __lhs, const _Pointer_adapter<_Tp> &__rhs)`
  - `template<typename _Tp>`  
`bool operator==(const _Pointer_adapter<_Tp> &__lhs, const _Pointer_adapter<_Tp> &__rhs)`
  - `template<size_t _Dimen, typename _RealType>`  
`bool operator==(const __gnu_cxx::normal_mv_distribution<_Dimen, _RealType> &__d1, const __gnu_cxx::normal_mv_distribution<_Dimen, _RealType> &__d2)`
  - `template<typename _Cond>`  
`bool operator==(const throw_value_base<_Cond> &__a, const throw_value_base<_Cond> &__b)`
  - `template<typename _Tp, typename _Poolp>`  
`bool operator==(const __mt_alloc<_Tp, _Poolp> &, const __mt_alloc<_Tp, _Poolp> &)`
  - `template<class _Tp, class _Alloc>`  
`bool operator==(const slist<_Tp, _Alloc> &__SL1, const slist<_Tp, _Alloc> &__SL2)`
  - `template<typename _Tp, typename _Cond>`  
`bool operator==(const throw_allocator_base<_Tp, _Cond> &, const throw_allocator_base<_Tp, _Cond> &)`
  - `template<typename _IteratorL, typename _IteratorR, typename _Container>`  
`constexpr bool operator==(const __normal_iterator<_IteratorL, _Container> &__lhs, const __normal_iterator<_IteratorR, _Container> &__rhs) noexcept`
  - `template<typename _Iterator, typename _Container>`  
`constexpr bool operator==(const __normal_iterator<_Iterator, _Container> &__lhs, const __normal_iterator<_Iterator, _Container> &__rhs) noexcept`
  - `template<typename _Tp1, typename _Tp2>`  
`bool operator==(const bitmap_allocator<_Tp1> &, const bitmap_allocator<_Tp2> &) throw ()`
  - `template<typename _CharT, typename _Traits, typename _Alloc, template<typename, typename> class _Base>`  
`bool operator==(const __versa_string<_CharT, _Traits, _Alloc, _Base> &__lhs, const __versa_string<_CharT, _Traits, _Alloc, _Base> &__rhs)`

- `template<typename _CharT, template< typename, typename, typename > class _Base>  
__enable_if< std::is_char< _CharT >::value, bool >::type operator== (const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__lhs, const __versa_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT >, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<class _CharT, class _Alloc >  
bool operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator> ( _Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >  
bool operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >  
bool operator> (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >  
bool operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >  
constexpr bool operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >  
bool operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >  
bool operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >  
bool operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator>= ( _Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >  
bool operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp >  
bool operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<class _Tp, class _Alloc >  
bool operator>= (const slist< _Tp, _Alloc > &__SL1, const slist< _Tp, _Alloc > &__SL2)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >  
bool operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`

- `template<typename _Iterator, typename _Container >`  
`constexpr bool operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`bool operator>= (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<size_t __Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< __Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`

- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::hypergeometric_distribution< _UIntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, logistic_distribution< _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > & __x)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp power (_Tp __x, _Integer __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator & __rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator & __rand)`
- `void rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`
- `template<typename _Tp >`  
`void swap (_ExtPtr_allocator< _Tp > & __larg, _ExtPtr_allocator< _Tp > & __rarg)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > & __hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > & __hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm1, hash_map< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm2)`
- `template<class _Val, class _HashFcn, class _EqualKey, class _Alloc >`  
`void swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > & __hs2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >`  
`void swap (hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm1, hash_multimap< _Key, _Tp, _HashFn, _EqKey, _Alloc > & __hm2)`
- `template<typename _Cond >`  
`void swap (throw_value_base< _Cond > & __a, throw_value_base< _Cond > & __b)`
- `template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All >`  
`void swap (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > & __ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > & __ht2)`
- `template<class _Tp, class _Alloc >`  
`void swap (slist< _Tp, _Alloc > & __x, slist< _Tp, _Alloc > & __y)`

- `template<class _CharT, class __Alloc >`  
`void swap (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<class _CharT, class _Alloc >`  
`void swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`std::pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __↵  
result)`

## Variables

- `static const _Lock_policy __default_lock_policy`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

### 3.1.1 Detailed Description

GNU extensions for public use.

### 3.1.2 Typedef Documentation

#### 3.1.2.1 `__int_traits`

```
template<typename _Tp >
using __gnu_cxx::__int_traits = typedef __numeric_traits_integer<_Tp>
```

Convenience alias for `__numeric_traits<integer-type>`.

Definition at line 136 of file `numeric_traits.h`.

### 3.1.3 Function Documentation

#### 3.1.3.1 `__static_pointer_cast()` [1/2]

```
template<typename _ToType, typename _FromType >
_ToType __gnu_cxx::__static_pointer_cast (
 const _FromType & __arg) [inline]
```

Casting operations for cases where `_FromType` is not a standard pointer. `_ToType` can be a standard or non-standard pointer. Given that `_FromType` is not a pointer, it must have a `get()` method that returns the standard pointer equivalent of the address it points to, and must have an `element_type` typedef which names the type it points to.

Definition at line 68 of file `cast.h`.

### 3.1.3.2 `__static_pointer_cast()` [2/2]

```
template<typename _ToType , typename _FromType >
_ToType __gnu_cxx::__static_pointer_cast (
 _FromType * __arg) [inline]
```

Casting operations for cases where `_FromType` is a standard pointer. `_ToType` can be a standard or non-standard pointer.

Definition at line 96 of file `cast.h`.

### 3.1.3.3 `_Bit_scan_forward()`

```
std::size_t __gnu_cxx::_Bit_scan_forward (
 std::size_t __num) [inline]
```

Generic Version of the `bsf` instruction.

Definition at line 508 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

### 3.1.3.4 `operator!=()` [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of two strings.

#### Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2389 of file `vstring.h`.



### 3.1.3.5 operator!=(()) [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!=(
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test difference of C string and string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

#### Returns

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 2402 of file `vstring.h`.

### 3.1.3.6 operator!=(()) [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator!=(
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test difference of string and C string.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

#### Returns

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 2415 of file `vstring.h`.

### 3.1.3.7 `operator+()` [1/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate two strings.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 181 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< ↵  
_CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

### 3.1.3.8 `operator+()` [2/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate C string and string.

#### Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

#### Returns

New string with value of `__lhs` followed by `__rhs`.

Definition at line 194 of file `vstring.tcc`.

## 3.1.3.9 operator+() [3/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 _CharT __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate character and string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 211 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

## 3.1.3.10 operator+() [4/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs)
```

Concatenate string and C string.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 224 of file `vstring.tcc`.

3.1.3.11 `operator+()` [5/5]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 _CharT __rhs)
```

Concatenate string and character.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

## Returns

New string with `__lhs` followed by `__rhs`.

Definition at line 241 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

3.1.3.12 `operator<()` [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string precedes string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2429 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

## 3.1.3.13 operator&lt;() [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string precedes C string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2442 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

## 3.1.3.14 operator&lt;() [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator< (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string precedes string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

## Returns

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 2455 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.15 operator<=()** [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't follow string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2509 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.16 operator<=()** [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't follow C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2522 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.17 operator<=()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator<= (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't follow string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 2535 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.18 operator==(** [1/4]

```
template<typename _Tp >
bool __gnu_cxx::operator==(
 const _Pointer_adapter< _Tp > & __lhs,
 const _Pointer_adapter< _Tp > & __rhs) [inline]
```

Comparison operators for `_Pointer_adapter` defer to the base class' comparison operators, when possible.

Definition at line 546 of file `pointer.h`.

**3.1.3.19 operator==(** [2/4]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator==(
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2338 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.20 operator==( ) [ 3 / 4 ]**

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator==(
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test equivalence of C string and string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 2362 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.21 operator==( ) [ 4 / 4 ]**

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator==(
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test equivalence of string and C string.



## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 2375 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.3.22 `operator>()` [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string follows string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

## Returns

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2469 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`.

3.1.3.23 `operator>()` [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string follows C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2482 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.24 `operator>()`** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator> (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string follows string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 2495 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.25 `operator>=()`** [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if string doesn't precede string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2549 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.26 operator>=()** [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't precede C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2562 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.27 operator>=()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
bool __gnu_cxx::operator>= (
 const _CharT * __lhs,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Test if C string doesn't precede string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 2575 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare()`.

**3.1.3.28 swap()**

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
void __gnu_cxx::swap (
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs) [inline]
```

Swap contents of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 2589 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

**3.2 \_\_gnu\_cxx::\_\_detail Namespace Reference****Classes**

- class `__mini_vector`
- class `__Bitmap_counter`
- class `__Ffit_finder`

**Enumerations**

- enum { `_S_max_rope_depth` }
- enum { `bits_per_byte`, `bits_per_block` }
- enum `_Tag` { `_S_leaf`, `_S_concat`, `_S_substringfn`, `_S_function` }

## Functions

- void `__bit_allocate` (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- void `__bit_free` (std::size\_t \* \_\_pmap, std::size\_t \_\_pos) throw ()
- template<typename \_ForwardIterator, typename \_Tp, typename \_Compare >  
\_ForwardIterator `__lower_bound` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp & \_\_val, \_Compare \_\_comp)
- template<typename \_AddrPair >  
std::size\_t `__num_bitmaps` (\_AddrPair \_\_ap)
- template<typename \_AddrPair >  
std::size\_t `__num_blocks` (\_AddrPair \_\_ap)

## 3.2.1 Detailed Description

Implementation details not part of the namespace `__gnu_cxx` interface.

## 3.2.2 Function Documentation

3.2.2.1 `__bit_allocate()`

```
void __gnu_cxx::__detail::__bit_allocate (
 std::size_t * __pmap,
 std::size_t __pos) throw () [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.

Definition at line 487 of file `bitmap_allocator.h`.

Referenced by `__gnu_cxx::bitmap_allocator<_Tp>::_M_allocate_single_object()`.

3.2.2.2 `__bit_free()`

```
void __gnu_cxx::__detail::__bit_free (
 std::size_t * __pmap,
 std::size_t __pos) throw () [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.

Definition at line 498 of file `bitmap_allocator.h`.

### 3.2.2.3 \_\_num\_bitmaps()

```
template<typename _AddrPair >
std::size_t __gnu_cxx::__detail::__num_bitmaps (
 _AddrPair __ap) [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.

Definition at line 274 of file bitmap\_allocator.h.

References [\\_\\_num\\_blocks\(\)](#).

Referenced by [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>::\\_M\\_allocate\\_single\\_object\(\)](#).

### 3.2.2.4 \_\_num\_blocks()

```
template<typename _AddrPair >
std::size_t __gnu_cxx::__detail::__num_blocks (
 _AddrPair __ap) [inline]
```

The number of Blocks pointed to by the address pair passed to the function.

Definition at line 266 of file bitmap\_allocator.h.

Referenced by [\\_\\_num\\_bitmaps\(\)](#).

## 3.3 \_\_gnu\_cxx::typelist Namespace Reference

### Functions

- [template<typename Fn , typename Typelist > void \*\*apply\*\* \(Fn &, Typelist\)](#)
- [template<typename Gn , typename Typelist > void \*\*apply\\_generator\*\* \(Gn &, Typelist\)](#)
- [template<typename Gn , typename TypelistT , typename TypelistV > void \*\*apply\\_generator\*\* \(Gn &, TypelistT, TypelistV\)](#)
- [template<typename Fn , typename Typelist > void \*\*apply\\_generator\*\* \(Fn &fn, Typelist\)](#)
- [template<typename Fn , typename TypelistT , typename TypelistV > void \*\*apply\\_generator\*\* \(Fn &fn, TypelistT, TypelistV\)](#)

### 3.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

### 3.3.2 Function Documentation

#### 3.3.2.1 `apply_generator()`

```
template<typename Gn , typename Typelist >
void __gnu_cxx::typelist::apply_generator (
 Gn & ,
 Typelist)
```

Apply all typelist types to generator functor.

## 3.4 `__gnu_debug` Namespace Reference

### Classes

- class [\\_After\\_nth\\_from](#)
- struct [\\_BeforeBeginHelper](#)
- class [\\_Equal\\_to](#)
- class [\\_Not\\_equal\\_to](#)
- class [\\_Safe\\_container](#)
- class [\\_Safe\\_forward\\_list](#)
- class [\\_Safe\\_iterator](#)
- class [\\_Safe\\_iterator\\_base](#)
- class [\\_Safe\\_local\\_iterator](#)
- class [\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_Safe\\_node\\_sequence](#)
- class [\\_Safe\\_sequence](#)
- class [\\_Safe\\_sequence\\_base](#)
- class [\\_Safe\\_unordered\\_container](#)
- class [\\_Safe\\_unordered\\_container\\_base](#)
- class [\\_Safe\\_vector](#)
- struct [\\_Sequence\\_traits](#)
- class [basic\\_string](#)

### Typedefs

- typedef [basic\\_string](#)< char > **string**
- typedef [basic\\_string](#)< wchar\_t > **wstring**

## Enumerations

- enum `_Debug_msg_id` {  
`__msg_valid_range`, `__msg_insert_singular`, `__msg_insert_different`, `__msg_erase_bad`,  
`__msg_erase_different`, `__msg_subscript_oob`, `__msg_empty`, `__msg_unpartitioned`,  
`__msg_unpartitioned_pred`, `__msg_unsorted`, `__msg_unsorted_pred`, `__msg_not_heap`,  
`__msg_not_heap_pred`, `__msg_bad_bitset_write`, `__msg_bad_bitset_read`, `__msg_bad_bitset_flip`,  
`__msg_self_splice`, `__msg_splice_alloc`, `__msg_splice_bad`, `__msg_splice_other`,  
`__msg_splice_overlap`, `__msg_init_singular`, `__msg_init_copy_singular`, `__msg_init_const_singular`,  
`__msg_copy_singular`, `__msg_bad_deref`, `__msg_bad_inc`, `__msg_bad_dec`,  
`__msg_iter_subscript_oob`, `__msg_advance_oob`, `__msg_retreat_oob`, `__msg_iter_compare_bad`,  
`__msg_compare_different`, `__msg_iter_order_bad`, `__msg_order_different`, `__msg_distance_bad`,  
`__msg_distance_different`, `__msg_deref_istream`, `__msg_inc_istream`, `__msg_output_ostream`,  
`__msg_deref_istreambuf`, `__msg_inc_istreambuf`, `__msg_insert_after_end`, `__msg_erase_after_bad`,  
`__msg_valid_range2`, `__msg_local_iter_compare_bad`, `__msg_non_empty_range`, `__msg_self_move` ←  
`assign`,  
`__msg_bucket_index_oob`, `__msg_valid_load_factor`, `__msg_equal_allocs`, `__msg_insert_range_from` ←  
`_self`,  
`__msg_irreflexive_ordering` }
- enum `_Distance_precision` {  
`__dp_none`, `__dp_equality`, `__dp_sign`, `__dp_sign_max_size`,  
`__dp_exact` }

## Functions

- `template<typename _Iterator, typename _Sequence >`  
`std::reverse_iterator< _Iterator > __base` (const `std::reverse_iterator< _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > > &__it`)
- `template<typename _Iterator >`  
`auto __base` (const `std::move_iterator< _Iterator > &__it`) -> `decltype(std::make_move_iterator(__base(__it. ←`  
`base()))`
- `template<typename _Iterator >`  
`constexpr _Iterator __base` (\_Iterator \_\_it)
- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __base` (const `_Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &__it`)
- `template<typename _Iterator, typename _Size >`  
`bool __can_advance` (const `std::reverse_iterator< _Iterator > &__it`, `_Size __n`)
- `template<typename _Iterator, typename _Size >`  
`bool __can_advance` (const `std::move_iterator< _Iterator > &__it`, `_Size __n`)
- `template<typename _InputIterator, typename _Size >`  
`constexpr bool __can_advance` (\_InputIterator, `_Size`)
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size >`  
`bool __can_advance` (const `_Safe_iterator< _Iterator, _Sequence, _Category > &`, `_Size`)
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool __check_partitioned_lower` (\_ForwardIterator \_\_first, `_ForwardIterator __last`, const `_Tp &__ ←`  
`value`)
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`constexpr bool __check_partitioned_lower` (\_ForwardIterator \_\_first, `_ForwardIterator __last`, const `_Tp &__ ←`  
`value`, `_Pred __pred`)
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool __check_partitioned_upper` (\_ForwardIterator \_\_first, `_ForwardIterator __last`, const `_Tp &__ ←`  
`value`)



- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`constexpr bool __check_partitioned_upper (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)`
- `template<typename _Iterator >`  
`bool __check_singular (const _Iterator &)`
- `template<typename _Tp >`  
`bool __check_singular (_Tp *const &__ptr)`
- `bool __check_singular_aux (const void *)`
- `bool __check_singular_aux (const __Safe_iterator_base * __x)`
- `template<typename _InputIterator >`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __check_sorted (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)`
- `template<typename _InputIterator >`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)`
- `template<typename _ForwardIterator >`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr bool __check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`constexpr bool __check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, std::__true_type)`
- `template<typename _InputIterator >`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::__false_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _Predicate, std::__false_type)`
- `template<typename _CharT, typename _Integer >`  
`const _CharT * __check_string (const _CharT * __s, _Integer __n, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT >`  
`const _CharT * __check_string (const _CharT * __s, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _InputIterator >`  
`_InputIterator __check_valid_range (const _InputIterator &__first, const _InputIterator &__last, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __foreign_iterator (const __Safe_iterator<_Iterator, _Sequence, _Category> &__it, _InputIterator __other, _InputIterator __other_end)`

- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral >`  
`bool __foreign_iterator_aux (const __Safe_iterator< _Iterator, _Sequence, _Category > &, _Integral, _Integral,`  
`std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __foreign_iterator_aux (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator`  
`__other, _InputIterator __other_end, std::__false_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const`  
`__Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const __Safe_iterator< _OtherIterator,`  
`_Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, type-`  
`name _OtherCategory >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const __Safe_iterator<`  
`_OtherIterator, _OtherSequence, _OtherCategory > &, const __Safe_iterator< _OtherIterator, _OtherSequence,`  
`_OtherCategory > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __foreign_iterator_aux2 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input<`  
`Iterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input<`  
`Iterator &__other, const _InputIterator &__other_end, std::__true_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __foreign_iterator_aux3 (const __Safe_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator`  
`&, const _InputIterator &, std::__false_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename`  
`_Sequence::value_type *__other)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __foreign_iterator_aux4 (const __Safe_iterator< _Iterator, _Sequence, _Category > &,...)`
- `template<typename _Iterator >`  
`__Distance_traits< _Iterator >::__type __get_distance (const std::reverse_iterator< _Iterator > &__first, const`  
`std::reverse_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`constexpr __Distance_traits< _Iterator >::__type __get_distance (_Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)`
- `template<typename _Iterator >`  
`__Distance_traits< _Iterator >::__type __get_distance (const std::move_iterator< _Iterator > &__first, const`  
`std::move_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`constexpr __Distance_traits< _Iterator >::__type __get_distance (_Iterator __lhs, _Iterator __rhs, std::input_iterator_tag)`
- `template<typename _Iterator >`  
`constexpr __Distance_traits< _Iterator >::__type __get_distance (_Iterator __lhs, _Iterator __rhs)`
- `template<typename _Iterator >`  
`constexpr bool __is_irreflexive (_Iterator __it)`
- `template<typename _Iterator, typename _Pred >`  
`constexpr bool __is_irreflexive_pred (_Iterator __it, _Pred __pred)`
- `template<typename _Iterator >`  
`auto __unsafe (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::make_reverse_iterator(__<`  
`unsafe(__it.base())))`
- `template<typename _Iterator >`  
`auto __unsafe (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__unsafe(__<`  
`__it.base())))`
- `template<typename _Iterator >`  
`_Iterator __unsafe (_Iterator __it)`

- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __unsafe (const \_Safe\_local\_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __unsafe (const \_Safe\_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator >`  
`bool __valid_range (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator >`  
`bool __valid_range (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __valid_range (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __valid_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > &, const \_Safe\_local\_iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range (_InputIterator __first, _InputIterator __last)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __valid_range (const \_Safe\_iterator< _Iterator, _Sequence, _Category > &, const \_Safe\_iterator< _Iterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __valid_range (const \_Safe\_local\_iterator< _Iterator, _Sequence > &, const \_Safe\_local\_iterator< _Iterator, _Sequence > &)`
- `template<typename _Integral >`  
`constexpr bool __valid_range_aux (_Integral, _Integral, std::true\_type)`
- `template<typename _Integral >`  
`constexpr bool __valid_range_aux (_Integral, _Integral, typename _Distance_traits< _Integral >::__type &__dist, std::true\_type)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::false\_type)`
- `template<typename _InputIterator >`  
`constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist, std::false\_type)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & getline (std::basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & getline (std::basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator!= (const _CharT *__lhs, const basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

- Generated by Doxygen

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool operator>= (const basic\_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void swap (basic\_string< _CharT, _Traits, _Allocator > &__lhs, basic\_string< _CharT, _Traits, _Allocator > &__rhs)`

### 3.4.1 Detailed Description

GNU debug classes for public use.

### 3.4.2 Enumeration Type Documentation

#### 3.4.2.1 `_Distance_precision`

```
enum __gnu_debug::_Distance_precision
```

The precision to which we can calculate the distance between two iterators.

Definition at line 52 of file `helper_functions.h`.

### 3.4.3 Function Documentation

#### 3.4.3.1 `__base()`

```
template<typename _Iterator >
constexpr _Iterator __gnu_debug::__base (
 _Iterator __it) [inline]
```

Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the `__valid_range` function thanks to the `<` operator.

Definition at line 299 of file `helper_functions.h`.

Referenced by `__gnu_parallel::__for_each_template_random_access_workstealing()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_before_dereferenceable()`, `std::boolalpha()`, `std::dec()`, `std::defaultfloat()`, `std::fixed()`, `std::hex()`, `std::hexfloat()`, `std::internal()`, `std::left()`, `std::noboolalpha()`, `std::noshowbase()`, `std::noshowpoint()`, `std::noshowpos()`, `std::noskipws()`, `std::nounitbuf()`, `std::nouppercase()`, `std::oct()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::right()`, `std::scientific()`, `std::showbase()`, `std::showpoint()`, `std::showpos()`, `std::skipws()`, `std::unitbuf()`, and `std::uppercase()`.

### 3.4.3.2 `__check_singular()`

```
template<typename _Tp >
bool __gnu_debug::__check_singular (
 _Tp *const & __ptr) [inline]
```

Non-NULL pointers are nonsingular.

Definition at line 130 of file `helper_functions.h`.

### 3.4.3.3 `__check_singular_aux()`

```
bool __gnu_debug::__check_singular_aux (
 const _Safe_iterator_base * __x) [inline]
```

Iterators that derive from `_Safe_iterator_base` can be determined singular or non-singular.

Definition at line 168 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_singular()`.

### 3.4.3.4 `__check_string()` [1/2]

```
template<typename _CharT , typename _Integer >
const _CharT* __gnu_debug::__check_string (
 const _CharT * __s,
 _Integer __n,
 const char * __file,
 unsigned int __line,
 const char * __function) [inline]
```

Checks that `__s` is non-NULL or `__n == 0`, and then returns `__s`.

Definition at line 49 of file `debug/string`.

### 3.4.3.5 `__check_string()` [2/2]

```
template<typename _CharT >
const _CharT* __gnu_debug::__check_string (
 const _CharT * __s,
 const char * __file,
 unsigned int __line,
 const char * __function) [inline]
```

Checks that `__s` is non-NULL and then returns `__s`.

Definition at line 65 of file `debug/string`.

## 3.4.3.6 \_\_foreign\_iterator\_aux2() [1/2]

```
template<typename _Iterator , typename _Sequence , typename _Category , typename _OtherIterator >
bool __gnu_debug::__foreign_iterator_aux2 (
 const __Safe_iterator< _Iterator, _Sequence, _Category > & __it,
 const __Safe_iterator< _OtherIterator, _Sequence, _Category > & __other,
 const __Safe_iterator< _OtherIterator, _Sequence, _Category > &) [inline]
```

Handle debug iterators from the same type of container.

Definition at line 127 of file functions.h.

## 3.4.3.7 \_\_foreign\_iterator\_aux2() [2/2]

```
template<typename _Iterator , typename _Sequence , typename _Category , typename _OtherIterator ,
typename _OtherSequence , typename _OtherCategory >
bool __gnu_debug::__foreign_iterator_aux2 (
 const __Safe_iterator< _Iterator, _Sequence, _Category > & ,
 const __Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ,
 const __Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &) [inline]
```

Handle debug iterators from different types of container.

Definition at line 138 of file functions.h.

## 3.4.3.8 \_\_get\_distance()

```
template<typename _Iterator >
constexpr _Distance_traits<_Iterator>::__type __gnu_debug::__get_distance (
 _Iterator __lhs,
 _Iterator __rhs,
 std::random_access_iterator_tag) [inline]
```

Determine the distance between two iterators with some known precision.

Definition at line 94 of file helper\_functions.h.

## 3.4.3.9 \_\_valid\_range() [1/3]

```
template<typename _InputIterator >
constexpr bool __gnu_debug::__valid_range (
 _InputIterator __first,
 _InputIterator __last,
 typename _Distance_traits< _InputIterator >::__type & __dist) [inline]
```

Don't know what these iterators are, or if they are even iterators (we may get an integral type for InputIterator), so see if they are integral and pass them on to the next phase otherwise.

Definition at line 225 of file helper\_functions.h.

References \_\_valid\_range\_aux().

**3.4.3.10 \_\_valid\_range()** [2/3]

```
template<typename _Iterator , typename _Sequence , typename _Category >
bool __gnu_debug::__valid_range (
 const __Safe_iterator< _Iterator, _Sequence, _Category > & __first,
 const __Safe_iterator< _Iterator, _Sequence, _Category > & __last,
 typename _Distance_traits< _Iterator >::__type & __dist) [inline]
```

Safe iterators know how to check if they form a valid range.

Definition at line 934 of file `safe_iterator.h`.

**3.4.3.11 \_\_valid\_range()** [3/3]

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::__valid_range (
 const __Safe_local_iterator< _Iterator, _Sequence > & __first,
 const __Safe_local_iterator< _Iterator, _Sequence > & __last,
 typename _Distance_traits< _Iterator >::__type & __dist_info) [inline]
```

Safe local iterators know how to check if they form a valid range.

Definition at line 407 of file `safe_local_iterator.h`.

**3.4.3.12 \_\_valid\_range\_aux()** [1/2]

```
template<typename _Integral >
constexpr bool __gnu_debug::__valid_range_aux (
 _Integral ,
 _Integral ,
 std::__true_type) [inline]
```

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.

Definition at line 140 of file `helper_functions.h`.

Referenced by `__valid_range()`, and `__valid_range_aux()`.

**3.4.3.13 \_\_valid\_range\_aux()** [2/2]

```
template<typename _InputIterator >
constexpr bool __gnu_debug::__valid_range_aux (
 _InputIterator __first,
 _InputIterator __last,
 std::__false_type) [inline]
```

We have iterators, so figure out what kind of iterators they are to see if we can check the range ahead of time.

Definition at line 181 of file `helper_functions.h`.

References `std::__iterator_category()`, and `__valid_range_aux()`.



## 3.5 `__gnu_internal` Namespace Reference

### 3.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

## 3.6 `__gnu_parallel` Namespace Reference

### Classes

- struct `__accumulate_binop_reduct`
- struct `__accumulate_selector`
- struct `__adjacent_difference_selector`
- struct `__adjacent_find_selector`
- class `__binder1st`
- class `__binder2nd`
- struct `__count_if_selector`
- struct `__count_selector`
- struct `__fill_selector`
- struct `__find_first_of_selector`
- struct `__find_if_selector`
- struct `__for_each_selector`
- struct `__generate_selector`
- struct `__generic_find_selector`
- struct `__generic_for_each_selector`
- struct `__identity_selector`
- struct `__inner_product_selector`
- struct `__max_element_reduct`
- struct `__min_element_reduct`
- struct `__mismatch_selector`
- struct `__multiway_merge_3_variant_sentinel_switch`
- struct `__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__multiway_merge_4_variant_sentinel_switch`
- struct `__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__multiway_merge_k_variant_sentinel_switch`
- struct `__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >`
- struct `__replace_if_selector`
- struct `__replace_selector`
- struct `__transform1_selector`
- struct `__transform2_selector`
- class `__unary_negate`
- struct `_DRandomShufflingGlobalData`
- struct `_DRSSorterPU`
- struct `_DummyReduct`
- class `_EqualFromLess`
- struct `_EqualTo`
- class `_GuardedIterator`
- class `_IteratorPair`

- class [\\_IteratorTriple](#)
- struct [\\_Job](#)
- struct [\\_Less](#)
- class [\\_Lexicographic](#)
- class [\\_LexicographicReverse](#)
- class [\\_LoserTree](#)
- class [\\_LoserTree< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeBase](#)
- class [\\_LoserTreePointer](#)
- class [\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerBase](#)
- class [\\_LoserTreePointerUnguarded](#)
- class [\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreePointerUnguardedBase](#)
- struct [\\_LoserTreeTraits](#)
- class [\\_LoserTreeUnguarded](#)
- class [\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#)
- class [\\_LoserTreeUnguardedBase](#)
- struct [\\_Multiplies](#)
- struct [\\_Nothing](#)
- struct [\\_Piece](#)
- struct [\\_Plus](#)
- struct [\\_PMWMSSortingData](#)
- class [\\_PseudoSequence](#)
- class [\\_PseudoSequenceIterator](#)
- struct [\\_QSBThreadLocal](#)
- class [\\_RandomNumber](#)
- class [\\_RestrictedBoundedConcurrentQueue](#)
- struct [\\_SamplingSorter](#)
- struct [\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#)
- struct [\\_Settings](#)
- struct [\\_SplitConsistently](#)
- struct [\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#)
- struct [balanced\\_quicksort\\_tag](#)
- struct [balanced\\_tag](#)
- struct [constant\\_size\\_blocks\\_tag](#)
- struct [default\\_parallel\\_tag](#)
- struct [equal\\_split\\_tag](#)
- struct [exact\\_tag](#)
- struct [find\\_tag](#)
- struct [growing\\_blocks\\_tag](#)
- struct [multiway\\_mergesort\\_exact\\_tag](#)
- struct [multiway\\_mergesort\\_sampling\\_tag](#)
- struct [multiway\\_mergesort\\_tag](#)
- struct [omp\\_loop\\_static\\_tag](#)
- struct [omp\\_loop\\_tag](#)
- struct [parallel\\_tag](#)
- struct [quicksort\\_tag](#)
- struct [sampling\\_tag](#)
- struct [sequential\\_tag](#)
- struct [unbalanced\\_tag](#)

## Typedefs

- typedef unsigned short [\\_BinIndex](#)
- typedef int64\_t [\\_CASable](#)
- typedef uint64\_t [\\_SequenceIndex](#)
- typedef uint16\_t [\\_ThreadIndex](#)

## Enumerations

- enum [\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_Parallelism](#) { **sequential**, **parallel\_unbalanced**, **parallel\_balanced**, **parallel\_omp\_loop**, **parallel\_omp\_loop\_static**, **parallel\_taskqueue** }
- enum [\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

## Functions

- template<typename \_Tp >  
\_Tp [\\_\\_add\\_omp](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- template<typename \_RAlter, typename \_DifferenceTp >  
void [\\_\\_calc\\_borders](#) (\_RAlter \_\_elements, \_DifferenceTp \_\_length, \_DifferenceTp \*\_\_off)
- template<typename \_Tp >  
bool [\\_\\_cas\\_omp](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Tp >  
bool [\\_\\_compare\\_and\\_swap](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_comparand, \_Tp \_\_replacement)
- template<typename \_Iter, typename \_OutputIterator >  
\_OutputIterator [\\_\\_copy\\_tail](#) (std::pair< \_Iter, \_Iter > \_\_b, std::pair< \_Iter, \_Iter > \_\_e, \_OutputIterator \_\_r)
- void [\\_\\_decode2](#) (\_CASable \_\_x, int &\_\_a, int &\_\_b)
- template<typename \_RAlter, typename \_DifferenceTp >  
void [\\_\\_determine\\_samples](#) (\_PMWMSortingData< \_RAlter > \*\_\_sd, \_DifferenceTp \_\_num\_samples)
- [\\_CASable](#) [\\_\\_encode2](#) (int \_\_a, int \_\_b)
- template<typename \_DifferenceType, typename \_OutputIterator >  
\_OutputIterator [\\_\\_equally\\_split](#) (\_DifferenceType \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType >  
\_DifferenceType [\\_\\_equally\\_split\\_point](#) (\_DifferenceType \_\_n, [\\_ThreadIndex](#) \_\_num\_threads, [\\_ThreadIndex](#) \_\_thread\_no)
- template<typename \_Tp >  
\_Tp [\\_\\_fetch\\_and\\_add](#) (volatile \_Tp \*\_\_ptr, \_Tp \_\_addend)
- template<typename \_RAlter1, typename \_RAlter2, typename \_Pred, typename \_Selector >  
std::pair< \_RAlter1, \_RAlter2 > [\\_\\_find\\_template](#) (\_RAlter1 \_\_begin1, \_RAlter1 \_\_end1, \_RAlter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector)
- template<typename \_RAlter1, typename \_RAlter2, typename \_Pred, typename \_Selector >  
std::pair< \_RAlter1, \_RAlter2 > [\\_\\_find\\_template](#) (\_RAlter1 \_\_begin1, \_RAlter1 \_\_end1, \_RAlter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [equal\\_split\\_tag](#))
- template<typename \_RAlter1, typename \_RAlter2, typename \_Pred, typename \_Selector >  
std::pair< \_RAlter1, \_RAlter2 > [\\_\\_find\\_template](#) (\_RAlter1 \_\_begin1, \_RAlter1 \_\_end1, \_RAlter2 \_\_begin2, \_Pred \_\_pred, \_Selector \_\_selector, [growing\\_blocks\\_tag](#))

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair< _RAIter1, _RAIter2 > __find_template ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`
- `template<typename _Iter, typename _UserOp, typename _Functionality, typename _Red, typename _Result >`  
`_UserOp __for_each_template_random_access ( _Iter __begin, _Iter __end, _UserOp __user_op, _Functionality & __functionality, _Red __reduction, _Result __reduction_start, _Result & __output, typename std::iterator_traits< _Iter >::difference_type __bound, _Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_ed ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_omp_loop_static ( _RAIter __begin, _RAIter __end, _Op __o, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >`  
`_Op __for_each_template_random_access_workstealing ( _RAIter __begin, _RAIter __end, _Op __op, _Fu & __f, _Red __r, _Result __base, _Result & __output, typename std::iterator_traits< _RAIter >::difference_type __bound)`
- `_ThreadIndex __get_max_threads ()`
- `bool __is_parallel (const _Parallelism __p)`
- `template<typename _Iter, typename _Compare >`  
`bool __is_sorted ( _Iter __begin, _Iter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __median_of_three_iterators ( _RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_movc ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >`  
`_OutputIterator __merge_advance_usual ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter2 & __begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare >`  
`_RAIter3 __parallel_merge_advance ( _RAIter1 & __begin1, _RAIter1 __end1, _RAIter1 & __begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_nth_element ( _RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_partial_sort ( _RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __parallel_partial_sum_basecase ( _Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _Iter >::value_type __value)`

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator\_traits<_Iter>::difference_type __n)`
- `template<typename _RAlter, typename _Predicate >`  
`std::iterator\_traits<_RAlter>::difference_type \_\_parallel\_partition (_RAlter __begin, _RAlter __end, _Predicate __pred, \_ThreadIndex __num_threads)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator __rng=\_RandomNumber())`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs (_RAlter __begin, _RAlter __end, typename std::iterator\_traits<_RAlter>::difference_type __n, \_ThreadIndex __num_threads, _RandomNumberGenerator &__rng)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void \_\_parallel\_random\_shuffle\_drs\_pu (\_DRSSorterPU<_RAlter, _RandomNumberGenerator> *__pus)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >`  
`_OutputIterator \_\_parallel\_set\_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_symmetric\_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >`  
`_OutputIterator \_\_parallel\_set\_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<bool __stable, typename _RAlter, typename _Compare, typename _Parallelism >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, \_Parallelism __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, multiway\_mergesort\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, multiway\_mergesort\_exact\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, balanced\_quicksort\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, default\_parallel\_tag __parallelism)`
- `template<bool __stable, typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort (_RAlter __begin, _RAlter __end, _Compare __comp, parallel\_tag __parallelism)`
- `template<typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort\_qs (_RAlter __begin, _RAlter __end, _Compare __comp, \_ThreadIndex __num_threads)`
- `template<typename _RAlter, typename _Compare >`  
`void \_\_parallel\_sort\_qs\_conquer (_RAlter __begin, _RAlter __end, _Compare __comp, \_ThreadIndex __num_threads)`

- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide ( _RAIter __begin, _RAIter __↵`  
`end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename`  
`std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __parallel_sort_qsb ( _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate >`  
`_OutputIterator __parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _Iter, class _OutputIterator >`  
`_OutputIterator __parallel_unique_copy ( _Iter __first, _Iter __last, _OutputIterator __result)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_conquer ( _QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __↵`  
`comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator_traits< _RAIter >::difference_type __qsb_divide ( _RAIter __begin, _RAIter __end, _Compare __↵`  
`comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void __qsb_local_sort_with_helping ( _QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex`  
`__iam, bool __wait)`
- `template<typename _RandomNumberGenerator >`  
`int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)`
- `template<typename _Size >`  
`_Size __rd_log2 ( _Size __n)`
- `template<typename _Tp >`  
`_Tp __round_up_to_pow2 ( _Tp __x)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >`  
`__RAIter1 __search_template ( __RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2,`  
`_Pred __pred)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare`  
`>`  
`_RAIter3 __sequential_multiway_merge ( _RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3`  
`__target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type↵`  
`::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIter, typename _RandomNumberGenerator >`  
`void __sequential_random_shuffle ( _RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)`
- `template<typename _Iter >`  
`void __shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length)`
- `template<typename _Iter >`  
`void __shrink_and_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_length,`  
`const bool __make_twice)`
- `void __yield ()`
- `template<typename _Iter, typename _FunctorType >`  
`size_t list_partition (const _Iter __begin, const _Iter __end, _Iter *__starts, size_t *__lengths, const int __↵`  
`num_parts, _FunctorType &__f, int __oversampling=0)`
- `template<typename _Tp >`  
`const _Tp &max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp >`  
`const _Tp &min (const _Tp &__a, const _Tp &__b)`
- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare >`  
`void multiseq_partition ( _RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator ↵`  
`__begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< ↵`  
`_RanSeqs >::value_type::first_type >::value_type >())`

- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare >`  
`_Tp multiway_selection` (`_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less<_Tp>()`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag` `__tag`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag` `__tag`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag` `__tag=parallel_tag(0)`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag` `__tag`)
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare >`  
`_RAlter3 multiway_merge_3_variant` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _DifferenceTp __length, _Compare __comp`)
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare >`  
`_RAlter3 multiway_merge_4_variant` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _DifferenceTp __length, _Compare __comp`)
- `template<bool __stable, typename _RAlterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway_merge_exact_splitting` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces`)
- `template<typename _LT, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare >`  
`_RAlter3 multiway_merge_loser_tree` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _DifferenceTp __length, _Compare __comp`)
- `template<typename _UnguardedLoserTree, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare >`  
`_RAlter3 multiway_merge_loser_tree_sentinel` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAlterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp`)
- `template<typename _LT, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare >`  
`_RAlter3 multiway_merge_loser_tree_unguarded` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAlterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp`)
- `template<bool __stable, typename _RAlterIterator, typename _Compare, typename _DifferenceType >`  
`void multiway_merge_sampling_splitting` (`_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge_sentinels` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag`)
- `template<typename _RAlterPairIterator, typename _RAlterOut, typename _DifferenceTp, typename _Compare >`  
`_RAlterOut multiway_merge_sentinels` (`_RAlterPairIterator __seqs_begin, _RAlterPairIterator __seqs_end, _RAlterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag` `__tag`)



- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↵`  
`_RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,`  
`typename _Compare >`  
`_RAIter3 parallel\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↵`  
`target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, ThreadIndex __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void parallel\_sort\_mwms\_pu (_PMWMSortingData< _RAIter > *__sd, _Compare &__comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R↵`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R↵`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R↵`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R↵`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _R↵`  
`AlterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel\_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↵`  
`seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

## Variables

- static const int [\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_CASable\\_mask](#)



### 3.6.1 Detailed Description

GNU parallel code for public use.

### 3.6.2 Typedef Documentation

#### 3.6.2.1 `_BinIndex`

```
typedef unsigned short __gnu_parallel::_BinIndex
```

Type to hold the index of a bin.

Since many variables of this type are allocated, it should be chosen as small as possible.

Definition at line 47 of file `random_shuffle.h`.

#### 3.6.2.2 `_CASable`

```
typedef int64_t __gnu_parallel::_CASable
```

Longest compare-and-swappable integer type on this platform.

Definition at line 127 of file `types.h`.

#### 3.6.2.3 `_SequenceIndex`

```
typedef uint64_t __gnu_parallel::_SequenceIndex
```

Unsigned integer to index `__elements`. The total number of elements for each algorithm must fit into this type.

Definition at line 117 of file `types.h`.

#### 3.6.2.4 `_ThreadIndex`

```
typedef uint16_t __gnu_parallel::_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

Definition at line 123 of file `types.h`.

### 3.6.3 Enumeration Type Documentation

#### 3.6.3.1 `_AlgorithmStrategy`

```
enum __gnu_parallel::_AlgorithmStrategy
```

Strategies for run-time algorithm selection:

Definition at line 67 of file types.h.

#### 3.6.3.2 `_FindAlgorithm`

```
enum __gnu_parallel::_FindAlgorithm
```

Find algorithms:

Definition at line 106 of file types.h.

#### 3.6.3.3 `_MultiwayMergeAlgorithm`

```
enum __gnu_parallel::_MultiwayMergeAlgorithm
```

Merging algorithms:

Definition at line 85 of file types.h.

#### 3.6.3.4 `_Parallelism`

```
enum __gnu_parallel::_Parallelism
```

Run-time equivalents for the compile-time tags.

##### Enumerator

|                                       |                                              |
|---------------------------------------|----------------------------------------------|
| <code>sequential</code>               | Not parallel.                                |
| <code>parallel_unbalanced</code>      | Parallel unbalanced (equal-sized chunks).    |
| <code>parallel_balanced</code>        | Parallel balanced (work-stealing).           |
| <code>parallel_omp_loop</code>        | Parallel with OpenMP dynamic load-balancing. |
| <code>parallel_omp_loop_static</code> | Parallel with OpenMP static load-balancing.  |
| <code>parallel_taskqueue</code>       | Parallel with OpenMP taskqueue construct.    |

Definition at line 44 of file types.h.

### 3.6.3.5 \_\_PartialSumAlgorithm

```
enum __gnu_parallel::__PartialSumAlgorithm
```

Partial sum algorithms: recursive, linear.

Definition at line 91 of file types.h.

### 3.6.3.6 \_\_SortAlgorithm

```
enum __gnu_parallel::__SortAlgorithm
```

Sorting algorithms:

Definition at line 76 of file types.h.

### 3.6.3.7 \_\_SplittingAlgorithm

```
enum __gnu_parallel::__SplittingAlgorithm
```

Sorting/merging algorithms: sampling, \_\_exact.

Definition at line 98 of file types.h.

## 3.6.4 Function Documentation

### 3.6.4.1 \_\_calc\_borders()

```
template<typename _RAIter , typename _DifferenceTp >
void __gnu_parallel::__calc_borders (
 _RAIter __elements,
 _DifferenceTp __length,
 _DifferenceTp * __off)
```

Precalculate \_\_advances for Knuth-Morris-Pratt algorithm.

#### Parameters

|                         |                                           |
|-------------------------|-------------------------------------------|
| <code>__elements</code> | Begin iterator of sequence to search for. |
| <code>__length</code>   | Length of sequence to search for.         |
| <code>__off</code>      | Returned __offsets.                       |

Definition at line 51 of file search.h.

Referenced by `__search_template()`.

### 3.6.4.2 `__compare_and_swap()`

```
template<typename _Tp >
bool __gnu_parallel::__compare_and_swap (
 volatile _Tp * __ptr,
 _Tp __comparand,
 _Tp __replacement) [inline]
```

Compare-and-swap.

Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return true, return false otherwise.

#### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__ptr</code>         | Pointer to signed integer. |
| <code>__comparand</code>   | Compare value.             |
| <code>__replacement</code> | Replacement value.         |

Definition at line 108 of file parallel/compatibility.h.

Referenced by `__parallel_partition()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > ::pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > > ::pop_↵front()`.

### 3.6.4.3 `__decode2()`

```
void __gnu_parallel::__decode2 (
 _CASable __x,
 int & __a,
 int & __b) [inline]
```

Decode two integers from one `gnu_parallel::_CASable`.

#### Parameters

|                        |                                                                                                                |
|------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>↵<br/>__x</code> | <code>__gnu_parallel::_CASable</code> to decode integers from.                                                 |
| <code>↵<br/>__a</code> | First integer, to be decoded from the most-significant <code>_CASable_bits/2</code> bits of <code>__x</code> . |
| <code>↵<br/>__b</code> | Second integer, to be encoded in the least-significant <code>_CASable_bits/2</code> bits of <code>__x</code> . |

See also

[\\_\\_encode2](#)

Definition at line 133 of file base.h.

References [\\_CASable\\_bits](#), and [\\_CASable\\_mask](#).

Referenced by [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::pop\\_back\(\)](#), [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::pop\\_front\(\)](#), and [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< pair< \\_RAIter, \\_RAIter > >::push\\_front\(\)](#).

#### 3.6.4.4 \_\_determine\_samples()

```
template<typename _RAIter , typename _DifferenceTp >
void __gnu_parallel::__determine_samples (
 _PMWSSortingData< _RAIter > * __sd,
 _DifferenceTp __num_samples)
```

Select [\\_M\\_samples](#) from a sequence.

Parameters

|                               |                                                                                           |
|-------------------------------|-------------------------------------------------------------------------------------------|
| <a href="#">__sd</a>          | Pointer to algorithm data. Result will be placed in <a href="#">__sd-&gt;_M_samples</a> . |
| <a href="#">__num_samples</a> | Number of <a href="#">_M_samples</a> to select.                                           |

Definition at line 97 of file multiway\_mergesort.h.

References [\\_\\_equally\\_split\(\)](#), [\\_\\_gnu\\_parallel::\\_PMWSSortingData< \\_RAIter >::\\_M\\_samples](#), [\\_\\_gnu\\_parallel::\\_PMWSSortingData< \\_RAIter >::\\_M\\_source](#), and [\\_\\_gnu\\_parallel::\\_PMWSSortingData< \\_RAIter >::\\_M\\_starts](#).

#### 3.6.4.5 \_\_encode2()

```
_CASable __gnu_parallel::__encode2 (
 int __a,
 int __b) [inline]
```

Encode two integers into one [gnu\\_parallel::\\_CASable](#).

Parameters

|                     |                                                                                              |
|---------------------|----------------------------------------------------------------------------------------------|
| <a href="#">__a</a> | First integer, to be encoded in the most-significant <a href="#">_CASable_bits/2</a> bits.   |
| <a href="#">__b</a> | Second integer, to be encoded in the least-significant <a href="#">_CASable_bits/2</a> bits. |

**Returns**

value encoding `__a` and `__b`.

**See also**

`__decode2`

Definition at line 119 of file `base.h`.

References `_CASable_bits`.

Referenced by `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_RestrictedBoundedConcurrentQueue()`, `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_pop_back()`, and `__gnu_parallel::_RestrictedBoundedConcurrentQueue< pair< _RAIter, _RAIter > >::_pop_front()`.

**3.6.4.6 `__equally_split()`**

```
template<typename _DifferenceType , typename _OutputIterator >
_OutputIterator __gnu_parallel::__equally_split (
 _DifferenceType __n,
 _ThreadIndex __num_threads,
 _OutputIterator __s)
```

function to split a sequence into parts of almost equal size.

The resulting sequence `__s` of length `__num_threads+1` contains the splitting positions when splitting the range `[0,__n)` into parts of almost equal size (plus minus 1). The first entry is 0, the last one `n`. There may result empty parts.

**Parameters**

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__s</code>           | Splitters          |

**Returns**

End of `__splitter` sequence, i.e. `__s+__num_threads+1`

Definition at line 48 of file `equally_split.h`.

Referenced by `__determine_samples()`, `__find_template()`, `__parallel_partial_sum_linear()`, `__parallel_unique_copy()`, and `__search_template()`.

## 3.6.4.7 \_\_equally\_split\_point()

```
template<typename _DifferenceType >
_DifferenceType __gnu_parallel::__equally_split_point (
 _DifferenceType __n,
 _ThreadIndex __num_threads,
 _ThreadIndex __thread_no)
```

function to split a sequence into parts of almost equal size.

Returns the position of the splitting point between thread number `__thread_no` (included) and thread number `__thread_no+1` (excluded).

## Parameters

|                            |                    |
|----------------------------|--------------------|
| <code>__n</code>           | Number of elements |
| <code>__num_threads</code> | Number of parts    |
| <code>__thread_no</code>   | Number of threads  |

## Returns

splitting point

Definition at line 75 of file `equally_split.h`.

Referenced by `__for_each_template_random_access_ed()`.

## 3.6.4.8 \_\_fetch\_and\_add()

```
template<typename _Tp >
_Tp __gnu_parallel::__fetch_and_add (
 volatile _Tp * __ptr,
 _Tp __addend) [inline]
```

Add a value to a variable, atomically.

## Parameters

|                       |                              |
|-----------------------|------------------------------|
| <code>__ptr</code>    | Pointer to a signed integer. |
| <code>__addend</code> | Value to add.                |

Definition at line 74 of file `parallel/compatibility.h`.

Referenced by `__parallel_partition()`.

3.6.4.9 `__find_template()` [1/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector) [inline]
```

Parallel `std::find`, switch for different algorithms.

## Parameters

|                         |                                                                                     |
|-------------------------|-------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                   |
| <code>__end1</code>     | End iterator of first sequence.                                                     |
| <code>__begin2</code>   | Begin iterator of second sequence. Must have same length as first sequence.         |
| <code>__pred</code>     | Find predicate.                                                                     |
| <code>__selector</code> | _Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...) |

## Returns

Place of finding in both sequences.

Definition at line 60 of file `find.h`.

References `__gnu_parallel::_Settings::get()`.

3.6.4.10 `__find_template()` [2/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 equal_split_tag)
```

Parallel `std::find`, equal splitting variant.

## Parameters

|                         |                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                             |
| <code>__end1</code>     | End iterator of first sequence.                                                               |
| <code>__begin2</code>   | Begin iterator of second sequence. Second __sequence must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                               |
| <code>__selector</code> | _Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |



**Returns**

Place of finding in both sequences.

Definition at line 97 of file find.h.

References `__equally_split()`, and `_GLIBCXX_CALL`.

**3.6.4.11 \_\_find\_template()** [3/4]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 growing_blocks_tag)
```

Parallel `std::find`, growing block size variant.

**Parameters**

|                         |                                                                                               |
|-------------------------|-----------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                             |
| <code>__end1</code>     | End iterator of first sequence.                                                               |
| <code>__begin2</code>   | Begin iterator of second sequence. Second __sequence must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                               |
| <code>__selector</code> | _Functionality (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

**Returns**

Place of finding in both sequences.

**See also**

`__gnu_parallel::_Settings::find_sequential_search_size`  
`__gnu_parallel::_Settings::find_scale_factor`

There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 185 of file find.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_scale_factor`, `__gnu_parallel::_Settings::find_sequential_search_size`, `std::pair<_T1, _T2>::first`, and `__gnu_parallel::_Settings::get()`.

### 3.6.4.12 `__find_template()` [ 4 / 4 ]

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred , typename _Selector >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_template (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred,
 _Selector __selector,
 constant_size_blocks_tag)
```

Parallel `std::find`, constant block size variant.

#### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <code>__begin1</code>   | Begin iterator of first sequence.                                                                          |
| <code>__end1</code>     | End iterator of first sequence.                                                                            |
| <code>__begin2</code>   | Begin iterator of second sequence. Second <code>__sequence</code> must have same length as first sequence. |
| <code>__pred</code>     | Find predicate.                                                                                            |
| <code>__selector</code> | <code>_Functionality</code> (e. g. <code>std::find_if()</code> , <code>std::equal()</code> ,...)           |

#### Returns

Place of finding in both sequences.

#### See also

`__gnu_parallel::_Settings::find_sequential_search_size`

`__gnu_parallel::_Settings::find_block_size` There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

Definition at line 315 of file `find.h`.

References `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::find_initial_block_size`, `__gnu_parallel::_Settings::find_sequential_search_size`, `std::pair<_T1, _T2>::first`, and `__gnu_parallel::_Settings::get()`.

### 3.6.4.13 `__for_each_template_random_access()`

```
template<typename _IIter , typename _UserOp , typename _Functionality , typename _Red , typename
_Result >
_UserOp __gnu_parallel::__for_each_template_random_access (
 _IIter __begin,
 _IIter __end,
 _UserOp __user_op,
```

```

_Functionality & __functionality,
_Red __reduction,
_Result __reduction_start,
_Result & __output,
typename std::iterator_traits< _Iter >::difference_type __bound,
_Parallelism __parallelism_tag)

```

Chose the desired algorithm by evaluating `__parallelism_tag`.

#### Parameters

|                                |                                                                                                                                                   |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>           | Begin iterator of input sequence.                                                                                                                 |
| <code>__end</code>             | End iterator of input sequence.                                                                                                                   |
| <code>__user_op</code>         | A user-specified functor (comparator, predicate, associative operator,...)                                                                        |
| <code>__functionality</code>   | functor to <i>process</i> an element with <code>__user_op</code> (depends on desired functionality, e. g. accumulate, <code>for_each</code> ,...) |
| <code>__reduction</code>       | Reduction functor.                                                                                                                                |
| <code>__reduction_start</code> | Initial value for reduction.                                                                                                                      |
| <code>__output</code>          | Output iterator.                                                                                                                                  |
| <code>__bound</code>           | Maximum number of elements processed.                                                                                                             |
| <code>__parallelism_tag</code> | Parallelization method                                                                                                                            |

Definition at line 61 of file `for_each.h`.

References `__for_each_template_random_access_ed()`, `__for_each_template_random_access_omp_loop()`, `__for_each_template_random_access_workstealing()`, `parallel_omp_loop`, `parallel_omp_loop_static`, and `parallel_unbalanced`.

#### 3.6.4.14 `__for_each_template_random_access_ed()`

```

template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_ed (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)

```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

#### Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__begin</code> | Begin iterator of element sequence. |
| <code>__end</code>   | End iterator of element sequence.   |

## Parameters

|                       |                                                                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...)                                                                       |
| <code>__f</code>      | Functor to "process" an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to "add" a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                |
| <code>__output</code> | Pointer to position where final result is written to                                                                                     |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                           |

## Returns

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file `par_loop.h`.

References `__equally_split_point()`.

Referenced by `__for_each_template_random_access()`.

3.6.4.15 `__for_each_template_random_access_omp_loop()`

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

## Parameters

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, etc.).                                                                          |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 67 of file omp\_loop.h.

Referenced by \_\_for\_each\_template\_random\_access().

**3.6.4.16 \_\_for\_each\_template\_random\_access\_omp\_loop\_static()**

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
 _RAIter __begin,
 _RAIter __end,
 _Op __o,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

**Parameters**

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__o</code>      | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed <code>__elements</code> (depends on functionality).             |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

**Returns**

User-supplied functor (that may contain a part of the result).

Definition at line 66 of file omp\_loop\_static.h.

**3.6.4.17 \_\_for\_each\_template\_random\_access\_workstealing()**

```
template<typename _RAIter , typename _Op , typename _Fu , typename _Red , typename _Result >
_Op __gnu_parallel::__for_each_template_random_access_workstealing (
 _RAIter __begin,
```

```

 _RAIter __end,
 _Op __op,
 _Fu & __f,
 _Red __r,
 _Result __base,
 _Result & __output,
 typename std::iterator_traits< _RAIter >::difference_type __bound)

```

Work stealing algorithm for random access iterators.

Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

#### Parameters

|                       |                                                                                                                                               |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>  | Begin iterator of element sequence.                                                                                                           |
| <code>__end</code>    | End iterator of element sequence.                                                                                                             |
| <code>__op</code>     | User-supplied functor (comparator, predicate, adding functor, ...).                                                                           |
| <code>__f</code>      | Functor to <i>process</i> an element with <code>__op</code> (depends on desired functionality, e. g. for <code>std::for_each()</code> , ...). |
| <code>__r</code>      | Functor to <i>add</i> a single <code>__result</code> to the already processed elements (depends on functionality).                            |
| <code>__base</code>   | Base value for reduction.                                                                                                                     |
| <code>__output</code> | Pointer to position where final result is written to                                                                                          |
| <code>__bound</code>  | Maximum number of elements processed (e. g. for <code>std::count_n()</code> ).                                                                |

#### Returns

User-supplied functor (that may contain a part of the result).

Definition at line 99 of file `workstealing.h`.

References `__gnu_debug::__base()`, `__yield()`, `_GLIBCXX_CALL`, `__gnu_parallel::Job< _DifferenceTp >::M_first`, `__gnu_parallel::Job< _DifferenceTp >::M_last`, `__gnu_parallel::Job< _DifferenceTp >::M_load`, `__gnu_parallel::Settings::cache_line_size`, `__gnu_parallel::Settings::get()`, and `min()`.

Referenced by `__for_each_template_random_access()`.

#### 3.6.4.18 `__is_sorted()`

```

template<typename _IIter , typename _Compare >
bool __gnu_parallel::__is_sorted (
 _IIter __begin,
 _IIter __end,
 _Compare __comp)

```

Check whether `[__begin, __end)` is sorted according to `__comp`.

#### Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__begin</code> | Begin iterator of sequence. |
| <code>__end</code>   | End iterator of sequence.   |
| <code>__comp</code>  | Comparator.                 |

**Returns**

`true` if sorted, `false` otherwise.

Definition at line 51 of file `checkers.h`.

**3.6.4.19 `__median_of_three_iterators()`**

```
template<typename _RAIter , typename _Compare >
_RAIter __gnu_parallel::__median_of_three_iterators (
 _RAIter __a,
 _RAIter __b,
 _RAIter __c,
 _Compare __comp)
```

Compute the median of three referenced elements, according to `__comp`.

**Parameters**

|                     |                  |
|---------------------|------------------|
| <code>__a</code>    | First iterator.  |
| <code>__b</code>    | Second iterator. |
| <code>__c</code>    | Third iterator.  |
| <code>__comp</code> | Comparator.      |

Definition at line 398 of file `base.h`.

**3.6.4.20 `__merge_advance()`**

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _Difference↵
Tp , typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp) [inline]
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence. |
|-----------------------|-----------------------------------|

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

Definition at line 171 of file merge.h.

References `__merge_advance_movc()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_merge_advance()`.

**3.6.4.21 `__merge_advance_movc()`**

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance_movc (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp)
```

Merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |



**Returns**

Output end iterator.

Definition at line 105 of file merge.h.

Referenced by \_\_merge\_advance().

**3.6.4.22 \_\_merge\_advance\_usual()**

```
template<typename _RAIter1 , typename _RAIter2 , typename _OutputIterator , typename _DifferenceTp , typename _Compare >
_OutputIterator __gnu_parallel::__merge_advance_usual (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter2 & __begin2,
 _RAIter2 __end2,
 _OutputIterator __target,
 _DifferenceTp __max_length,
 _Compare __comp)
```

Merge routine being able to merge only the \_\_max\_length smallest elements.

The \_\_begin iterators are advanced accordingly, they might not reach \_\_end, in contrast to the usual variant.

**Parameters**

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

**Returns**

Output end iterator.

Definition at line 57 of file merge.h.

**3.6.4.23 \_\_parallel\_merge\_advance()** [1/2]

```
template<typename _RAIter1 , typename _RAIter2 , typename _RAIter3 , typename _Compare >
_RAIter3 __gnu_parallel::__parallel_merge_advance (
```

```

_RAIter1 & __begin1,
_RAIter1 __end1,
_RAIter2 & __begin2,
_RAIter2 __end2,
_RAIter3 __target,
typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
_Compare __comp) [inline]

```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

#### Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__begin1</code>     | Begin iterator of first sequence.    |
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

#### Returns

Output end iterator.

Definition at line 195 of file merge.h.

References `__merge_advance()`.

#### 3.6.4.24 `__parallel_merge_advance()` [2/2]

```

template<typename _RAIter1 , typename _RAIter3 , typename _Compare >
_RAIter3 __gnu_parallel::__parallel_merge_advance (
 _RAIter1 & __begin1,
 _RAIter1 __end1,
 _RAIter1 & __begin2,
 _RAIter1 __end2,
 _RAIter3 __target,
 typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
 _Compare __comp) [inline]

```

Parallel merge routine being able to merge only the `__max_length` smallest elements.

The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto `parallel_multiway_merge`.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence. |
|-----------------------|-----------------------------------|

## Parameters

|                           |                                      |
|---------------------------|--------------------------------------|
| <code>__end1</code>       | End iterator of first sequence.      |
| <code>__begin2</code>     | Begin iterator of second sequence.   |
| <code>__end2</code>       | End iterator of second sequence.     |
| <code>__target</code>     | Target begin iterator.               |
| <code>__max_length</code> | Maximum number of elements to merge. |
| <code>__comp</code>       | Comparator.                          |

## Returns

Output end iterator.

Definition at line 223 of file `merge.h`.

References `multiway_merge_exact_splitting()`, and `parallel_multiway_merge()`.

3.6.4.25 `__parallel_nth_element()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_nth_element (
 _RAIter __begin,
 _RAIter __nth,
 _RAIter __end,
 _Compare __comp)
```

Parallel implementation of `std::nth_element()`.

## Parameters

|                      |                                                          |
|----------------------|----------------------------------------------------------|
| <code>__begin</code> | Begin iterator of input sequence.                        |
| <code>__nth</code>   | Iterator of element that must be in position afterwards. |
| <code>__end</code>   | End iterator of input sequence.                          |
| <code>__comp</code>  | Comparator.                                              |

Definition at line 332 of file `partition.h`.

References `__parallel_partition()`, `_GLIBCXX_CALL`, `__gnu_parallel::_Settings::get()`, `std::max()`, `__gnu_parallel::_Settings::nth_element_minimal_n`, and `__gnu_parallel::_Settings::partition_minimal_n`.

Referenced by `__parallel_partial_sort()`.

### 3.6.4.26 `__parallel_partial_sort()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_partial_sort (
 _RAIter __begin,
 _RAIter __middle,
 _RAIter __end,
 _Compare __comp)
```

Parallel implementation of `std::partial_sort()`.

#### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence. |
| <code>__middle</code> | Sort until this position.         |
| <code>__end</code>    | End iterator of input sequence.   |
| <code>__comp</code>   | Comparator.                       |

Definition at line 422 of file `partition.h`.

References `__parallel_nth_element()`.

### 3.6.4.27 `__parallel_partial_sum()`

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op)
```

Parallel partial sum front-`__end`.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin</code>  | Begin iterator of input sequence.  |
| <code>__end</code>    | End iterator of input sequence.    |
| <code>__result</code> | Begin iterator of output sequence. |
| <code>__bin_op</code> | Associative binary function.       |

#### Returns

End iterator of output sequence.

Definition at line 205 of file `partial_sum.h`.

References `__parallel_partial_sum_linear()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

## 3.6.4.28 \_\_parallel\_partial\_sum\_basecase()

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op,
 typename std::iterator_traits< _IIter >::value_type __value)
```

Base case prefix sum routine.

## Parameters

|                 |                                                                              |
|-----------------|------------------------------------------------------------------------------|
| <i>__begin</i>  | Begin iterator of input sequence.                                            |
| <i>__end</i>    | End iterator of input sequence.                                              |
| <i>__result</i> | Begin iterator of output sequence.                                           |
| <i>__bin_op</i> | Associative binary function.                                                 |
| <i>__value</i>  | Start value. Must be passed since the neutral element is unknown in general. |

## Returns

End iterator of output sequence.

Definition at line 58 of file partial\_sum.h.

Referenced by \_\_parallel\_partial\_sum\_linear().

## 3.6.4.29 \_\_parallel\_partial\_sum\_linear()

```
template<typename _IIter , typename _OutputIterator , typename _BinaryOperation >
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
 _IIter __begin,
 _IIter __end,
 _OutputIterator __result,
 _BinaryOperation __bin_op,
 typename std::iterator_traits< _IIter >::difference_type __n)
```

Parallel partial sum implementation, two-phase approach, no recursion.

## Parameters

|                 |                                    |
|-----------------|------------------------------------|
| <i>__begin</i>  | Begin iterator of input sequence.  |
| <i>__end</i>    | End iterator of input sequence.    |
| <i>__result</i> | Begin iterator of output sequence. |
| <i>__bin_op</i> | Associative binary function.       |
| <i>__n</i>      | Length of sequence.                |

**Returns**

End iterator of output sequence.

Definition at line 89 of file `partial_sum.h`.

References `__equally_split()`, `__parallel_partial_sum_basecase()`, `std::accumulate()`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::partial_sum_dilation`.

Referenced by `__parallel_partial_sum()`.

**3.6.4.30 \_\_parallel\_partition()**

```
template<typename _RAIter , typename _Predicate >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_partition (
 _RAIter __begin,
 _RAIter __end,
 _Predicate __pred,
 _ThreadIndex __num_threads)
```

Parallel implementation of `std::partition`.

**Parameters**

|                            |                                                             |
|----------------------------|-------------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence to split.                  |
| <code>__end</code>         | End iterator of input sequence to split.                    |
| <code>__pred</code>        | Partition predicate, possibly including some kind of pivot. |
| <code>__num_threads</code> | Maximum number of threads to use for this task.             |

**Returns**

Number of elements not fulfilling the predicate.

Definition at line 56 of file `partition.h`.

References `__compare_and_swap()`, `__fetch_and_add()`, `_GLIBCXX_CALL`, `_GLIBCXX_VOLATILE`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::partition_chunk_share`, and `__gnu_parallel::_Settings::partition_chunk_size`.

Referenced by `__parallel_nth_element()`.

**3.6.4.31 \_\_parallel\_random\_shuffle()**

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle (
 _RAIter __begin,
 _RAIter __end,
 _RandomNumberGenerator __rng = _RandomNumber()) [inline]
```

Parallel random public call.

## Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

Definition at line 522 of file `random_shuffle.h`.

References `__parallel_random_shuffle_drs()`.

## 3.6.4.32 \_\_parallel\_random\_shuffle\_drs()

```
template<typename _RAIter, typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle_drs (
 _RAIter __begin,
 _RAIter __end,
 typename std::iterator_traits< _RAIter >::difference_type __n,
 _ThreadIndex __num_threads,
 _RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

## Parameters

|                            |                                 |
|----------------------------|---------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.     |
| <code>__end</code>         | End iterator of sequence.       |
| <code>__n</code>           | Length of sequence.             |
| <code>__num_threads</code> | Number of threads to use.       |
| <code>__rng</code>         | Random number generator to use. |

Definition at line 265 of file `random_shuffle.h`.

References `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_bins_end`, `__parallel_random_shuffle_drs_pu()`, `__rd_log2()`, `__round_up_to_pow2()`, `__sequential_random_shuffle()`, `_GLIBCXX_CALL`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle()`.

### 3.6.4.33 `__parallel_random_shuffle_drs_pu()`

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
 _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus)
```

Random shuffle code executed by each thread.

#### Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pus</code> | Array of thread-local data records. |
|--------------------|-------------------------------------|

Definition at line 122 of file `random_shuffle.h`.

References `__random_number_pow2()`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_sd`, `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed`, `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts`, and `std::partial_sum()`.

Referenced by `__parallel_random_shuffle_drs()`.

### 3.6.4.34 `__parallel_sort()` [1/7]

```
template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_tag __parallelism) [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

#### Template Parameters

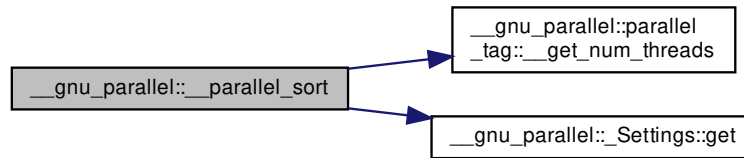
|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 75 of file `sort.h`.



References \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads(), \_GLIBCXX\_CALL, and \_\_gnu\_parallel::\_Settings::\_\_get().

Here is the call graph for this function:



#### 3.6.4.35 \_\_parallel\_sort() [2/7]

```

template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_exact_tag __parallelism) [inline]

```

Choose multiway mergesort with exact splitting, for parallel sorting.

##### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

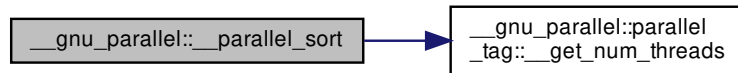
##### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 99 of file `sort.h`.

References \_\_gnu\_parallel::parallel\_tag::\_\_get\_num\_threads(), and \_GLIBCXX\_CALL.

Here is the call graph for this function:



### 3.6.4.36 \_\_parallel\_sort() [3/7]

```

template<bool __stable, typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 multiway_mergesort_sampling_tag __parallelism) [inline]

```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

#### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

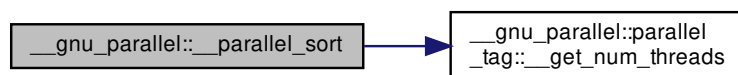
#### Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 120 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



**3.6.4.37 \_\_parallel\_sort()** [4/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 quicksort_tag __parallelism) [inline]
```

Choose quicksort for parallel sorting.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

**Template Parameters**

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 140 of file sort.h.

References `_GLIBCXX_CALL`.

**3.6.4.38 \_\_parallel\_sort()** [5/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 balanced_quicksort_tag __parallelism) [inline]
```

Choose balanced quicksort for parallel sorting.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

## Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 161 of file sort.h.

References `_GLIBCXX_CALL`.

3.6.4.39 `__parallel_sort()` [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 default_parallel_tag __parallelism) [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

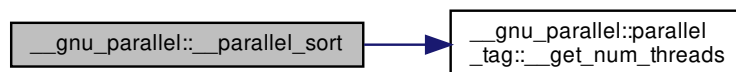
## Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 183 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, and `_GLIBCXX_CALL`.

Here is the call graph for this function:



## 3.6.4.40 \_\_parallel\_sort() [7/7]

```
template<bool __stable, typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 parallel_tag __parallelism) [inline]
```

Choose a parallel sorting algorithm.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__begin</code> | Begin iterator of input sequence. |
| <code>__end</code>   | End iterator of input sequence.   |
| <code>__comp</code>  | Comparator.                       |

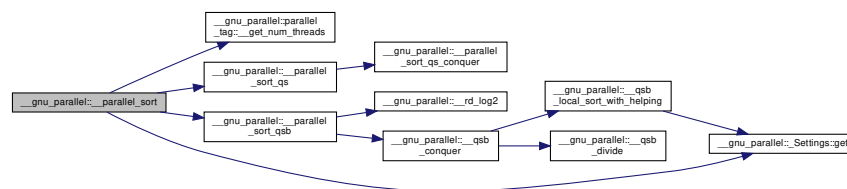
## Template Parameters

|                       |              |
|-----------------------|--------------|
| <code>__stable</code> | Sort stable. |
|-----------------------|--------------|

Definition at line 203 of file sort.h.

References `__gnu_parallel::parallel_tag::__get_num_threads()`, `__parallel_sort_qs()`, `__parallel_sort_qsb()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_Settings::get()`.

Here is the call graph for this function:



## 3.6.4.41 \_\_parallel\_sort\_qs()

```
template<typename _RAIter, typename _Compare >
void __gnu_parallel::__parallel_sort_qs (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort main call.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of input sequence.                        |
| <code>__end</code>         | End iterator input sequence, ignored.                    |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 154 of file quicksort.h.

References `__parallel_sort_qs_conquer()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_sort()`.

3.6.4.42 `__parallel_sort_qs_conquer()`

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort_qs_conquer (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort conquer step.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 101 of file quicksort.h.

Referenced by `__parallel_sort_qs()`.

3.6.4.43 `__parallel_sort_qs_divide()`

```
template<typename _RAIter , typename _Compare >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__parallel_sort_qs_divide (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 typename std::iterator_traits<_RAIter >::difference_type __pivot_rank,
 typename std::iterator_traits<_RAIter >::difference_type __num_samples,
 _ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__pivot_rank</code>  | Desired <code>__rank</code> of the pivot.                |
| <code>__num_samples</code> | Choose pivot from that many samples.                     |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 51 of file quicksort.h.

References `std::min()`.

## 3.6.4.44 \_\_parallel\_sort\_qsb()

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__parallel_sort_qsb (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Top-level quicksort routine.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of sequence.                              |
| <code>__end</code>         | End iterator of sequence.                                |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 433 of file balanced\_quicksort.h.

References `__qsb_conquer()`, `__rd_log2()`, `_GLIBCXX_CALL`, and `__gnu_parallel::_QSBThreadLocal<_RAIter>::↵_M_elements_leftover`.

Referenced by `__parallel_sort()`.

## 3.6.4.45 \_\_parallel\_unique\_copy() [1/2]

```
template<typename _IIter , class _OutputIterator , class _BinaryPredicate >
_OutputIterator __gnu_parallel::__parallel_unique_copy (
 _IIter __first,
 _IIter __last,
 _OutputIterator __result,
 _BinaryPredicate __binary_pred)
```

Parallel `std::unique_copy()`, w/`__o` explicit equality predicate.

**Parameters**

|                            |                                                    |
|----------------------------|----------------------------------------------------|
| <code>__first</code>       | Begin iterator of input sequence.                  |
| <code>__last</code>        | End iterator of input sequence.                    |
| <code>__result</code>      | Begin iterator of result <code>__sequence</code> . |
| <code>__binary_pred</code> | Equality predicate.                                |

**Returns**

End iterator of result `__sequence`.

Definition at line 50 of file `unique_copy.h`.

References `__equally_split()`, and `_GLIBCXX_CALL`.

Referenced by `__parallel_unique_copy()`.

**3.6.4.46 `__parallel_unique_copy()` [2/2]**

```
template<typename _IIter , class _OutputIterator >
_OutputIterator __gnu_parallel::__parallel_unique_copy (
 _IIter __first,
 _IIter __last,
 _OutputIterator __result) [inline]
```

Parallel `std::unique_copy()`, without explicit equality predicate.

**Parameters**

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <code>__first</code>  | Begin iterator of input sequence.                  |
| <code>__last</code>   | End iterator of input sequence.                    |
| <code>__result</code> | Begin iterator of result <code>__sequence</code> . |

**Returns**

End iterator of result `__sequence`.

Definition at line 186 of file `unique_copy.h`.

References `__parallel_unique_copy()`.



## 3.6.4.47 \_\_qsb\_conquer()

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__qsb_conquer (
 _QSBThreadLocal< _RAIter > ** __tls,
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __iam,
 _ThreadIndex __num_threads,
 bool __parent_wait)
```

Quicksort conquer step.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__tls</code>         | Array of thread-local storages.                          |
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__iam</code>         | Number of the thread processing this function.           |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

Definition at line 174 of file `balanced_quicksort.h`.

References `__qsb_divide()`, `__qsb_local_sort_with_helping()`, and `__gnu_parallel::_QSBThreadLocal< _RAIter >::M_initial`.

Referenced by `__parallel_sort_qsb()`.

## 3.6.4.48 \_\_qsb\_divide()

```
template<typename _RAIter , typename _Compare >
std::iterator_traits<_RAIter>::difference_type __gnu_parallel::__qsb_divide (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Balanced quicksort divide step.

## Parameters

|                            |                                                          |
|----------------------------|----------------------------------------------------------|
| <code>__begin</code>       | Begin iterator of subsequence.                           |
| <code>__end</code>         | End iterator of subsequence.                             |
| <code>__comp</code>        | Comparator.                                              |
| <code>__num_threads</code> | Number of threads that are allowed to work on this part. |

**Precondition**

`(__end-__begin)>=1`

Definition at line 103 of file `balanced_quicksort.h`.

Referenced by `__qsb_conquer()`.

**3.6.4.49 `__qsb_local_sort_with_helping()`**

```
template<typename _RAIter , typename _Compare >
void __gnu_parallel::__qsb_local_sort_with_helping (
 _QSBThreadLocal< _RAIter > ** __tls,
 _Compare & __comp,
 _ThreadIndex __iam,
 bool __wait)
```

Quicksort step doing load-balanced local sort.

**Parameters**

|                     |                                                |
|---------------------|------------------------------------------------|
| <code>__tls</code>  | Array of thread-local storages.                |
| <code>__comp</code> | Comparator.                                    |
| <code>__iam</code>  | Number of the thread processing this function. |

Definition at line 250 of file `balanced_quicksort.h`.

References `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_initial`, `__gnu_parallel::__QSBThreadLocal< _RAIter >::__M_num_threads`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::sort_qsb_base_case_maximal←_n`.

Referenced by `__qsb_conquer()`.

**3.6.4.50 `__random_number_pow2()`**

```
template<typename _RandomNumberGenerator >
int __gnu_parallel::__random_number_pow2 (
 int __logp,
 _RandomNumberGenerator & __rng) [inline]
```

Generate a random number in  $[0, 2^{\text{__logp}})$ .

**Parameters**

|                     |                                                               |
|---------------------|---------------------------------------------------------------|
| <code>__logp</code> | Logarithm (basis 2) of the upper range <code>__bound</code> . |
| <code>__rng</code>  | Random number generator to use.                               |

Definition at line 115 of file random\_shuffle.h.

Referenced by \_\_parallel\_random\_shuffle\_drs\_pu(), and \_\_sequential\_random\_shuffle().

#### 3.6.4.51 \_\_rd\_log2()

```
template<typename _Size >
_Size __gnu_parallel::__rd_log2 (
 _Size __n) [inline]
```

Calculates the rounded-down logarithm of \_\_n for base 2.

##### Parameters

|                   |           |
|-------------------|-----------|
| $\leftrightarrow$ | Argument. |
| <code>__n</code>  |           |

##### Returns

Returns 0 for any argument <1.

Definition at line 102 of file base.h.

Referenced by \_\_parallel\_random\_shuffle\_drs(), \_\_parallel\_sort\_qsb(), \_\_round\_up\_to\_pow2(), \_\_sequential\_↔random\_shuffle(), \_\_gnu\_parallel::\_LoserTreeBase< \_Tp, \_Compare >::\_LoserTreeBase(), and multiseq\_selection().

#### 3.6.4.52 \_\_round\_up\_to\_pow2()

```
template<typename _Tp >
_Tp __gnu_parallel::__round_up_to_pow2 (
 _Tp __x)
```

Round up to the next greater power of 2.

##### Parameters

|                   |                     |
|-------------------|---------------------|
| $\leftrightarrow$ | Integer to round up |
| <code>__x</code>  |                     |

Definition at line 248 of file random\_shuffle.h.

References \_\_rd\_log2().

Referenced by \_\_parallel\_random\_shuffle\_drs(), \_\_sequential\_random\_shuffle(), and multiseq\_selection().

### 3.6.4.53 `__search_template()`

```
template<typename __RAIter1 , typename __RAIter2 , typename _Pred >
__RAIter1 __gnu_parallel::__search_template (
 __RAIter1 __begin1,
 __RAIter1 __end1,
 __RAIter2 __begin2,
 __RAIter2 __end2,
 _Pred __pred)
```

Parallel `std::search`.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__end2</code>   | End iterator of second sequence.   |
| <code>__pred</code>   | Find predicate.                    |

#### Returns

Place of finding in first sequences.

Definition at line 81 of file `search.h`.

References `__calc_borders()`, `__equally_split()`, `_GLIBCXX_CALL`, and `std::min()`.

### 3.6.4.54 `__sequential_multiway_merge()`

```
template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename
_DifferenceTp , typename _Compare >
__RAIter3 __gnu_parallel::__sequential_multiway_merge (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)
```

Sequential multi-way merging switch.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

#### Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |
| <code>__sentinel</code>   | The sequences have a sentinel element.                                          |

**Returns**

End iterator of output sequence.

Definition at line 920 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

Referenced by `multiway_merge()`, and `multiway_merge_sentinels()`.

**3.6.4.55 \_\_sequential\_random\_shuffle()**

```
template<typename _RAIter , typename _RandomNumberGenerator >
void __gnu_parallel::__sequential_random_shuffle (
 _RAIter __begin,
 _RAIter __end,
 _RandomNumberGenerator & __rng)
```

Sequential cache-efficient random shuffle.

**Parameters**

|                      |                                 |
|----------------------|---------------------------------|
| <code>__begin</code> | Begin iterator of sequence.     |
| <code>__end</code>   | End iterator of sequence.       |
| <code>__rng</code>   | Random number generator to use. |

Definition at line 410 of file random\_shuffle.h.

References `__random_number_pow2()`, `__rd_log2()`, `__round_up_to_pow2()`, `__gnu_parallel::_Settings::get()`, `__gnu_parallel::_Settings::L2_cache_size`, `std::min()`, `std::partial_sum()`, and `__gnu_parallel::_Settings::TLB_size`.

Referenced by `__parallel_random_shuffle_drs()`.

**3.6.4.56 \_\_shrink()**

```
template<typename _IIter >
void __gnu_parallel::__shrink (
 std::vector<_IIter > & __os_starts,
 size_t & __count_to_two,
 size_t & __range_length)
```

Combines two ranges into one and thus halves the number of ranges.

**Parameters**

|                             |                                          |
|-----------------------------|------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled). |
| <code>__count_to_two</code> | Counts up to 2.                          |
| <code>__range_length</code> | Current length of a chunk.               |

Definition at line 70 of file list\_partition.h.

References `std::vector<_Tp, _Alloc >::size()`.

Referenced by `__shrink_and_double()`.

#### 3.6.4.57 `__shrink_and_double()`

```
template<typename _IIter >
void __gnu_parallel::__shrink_and_double (
 std::vector<_IIter > & __os_starts,
 size_t & __count_to_two,
 size_t & __range_length,
 const bool __make_twice)
```

Shrinks and doubles the ranges.

##### Parameters

|                             |                                                                    |
|-----------------------------|--------------------------------------------------------------------|
| <code>__os_starts</code>    | Start positions worked on (oversampled).                           |
| <code>__count_to_two</code> | Counts up to 2.                                                    |
| <code>__range_length</code> | Current length of a chunk.                                         |
| <code>__make_twice</code>   | Whether the <code>__os_starts</code> is allowed to be grown or not |

Definition at line 50 of file list\_partition.h.

References `__shrink()`, `std::vector<_Tp, _Alloc >::resize()`, and `std::vector<_Tp, _Alloc >::size()`.

Referenced by `list_partition()`.

#### 3.6.4.58 `__yield()`

```
void __gnu_parallel::__yield () [inline]
```

Yield control to another thread, without waiting for the end of the time slice.

Definition at line 121 of file parallel/compatibility.h.

Referenced by `__for_each_template_random_access_workstealing()`.

## 3.6.4.59 list\_partition()

```
template<typename _IIter , typename _FunctorType >
size_t __gnu_parallel::list_partition (
 const _IIter __begin,
 const _IIter __end,
 _IIter * __starts,
 size_t * __lengths,
 const int __num_parts,
 _FunctorType & __f,
 int __oversampling = 0)
```

Splits a sequence given by input iterators into parts of almost equal size.

The function needs only one pass over the sequence.

## Parameters

|                             |                                                                                                                                                                                                                                                   |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin</code>        | Begin iterator of input sequence.                                                                                                                                                                                                                 |
| <code>__end</code>          | End iterator of input sequence.                                                                                                                                                                                                                   |
| <code>__starts</code>       | Start iterators for the resulting parts, dimension <code>__num_parts+1</code> . For convenience, <code>__starts [ __num_parts ]</code> contains the end iterator of the sequence.                                                                 |
| <code>__lengths</code>      | Length of the resulting parts.                                                                                                                                                                                                                    |
| <code>__num_parts</code>    | Number of parts to split the sequence into.                                                                                                                                                                                                       |
| <code>__f</code>            | Functor to be applied to each element by traversing <code>__it</code>                                                                                                                                                                             |
| <code>__oversampling</code> | Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num\_parts})$ |

## Returns

Length of the whole sequence.

Definition at line 101 of file `list_partition.h`.

References `__shrink_and_double()`, and `std::vector< _Tp, _Alloc >::size()`.

## 3.6.4.60 max()

```
template<typename _Tp >
const _Tp& __gnu_parallel::max (
 const _Tp & __a,
 const _Tp & __b) [inline]
```

Equivalent to `std::max`.

Definition at line 150 of file `base.h`.

### 3.6.4.61 min()

```
template<typename _Tp >
const _Tp& __gnu_parallel::min (
 const _Tp & __a,
 const _Tp & __b) [inline]
```

Equivalent to `std::min`.

Definition at line 144 of file `base.h`.

Referenced by `__for_each_template_random_access_workstealing()`.

### 3.6.4.62 multiseq\_partition()

```
template<typename _RanSeqs , typename _RankType , typename _RankIterator , typename _Compare >
void __gnu_parallel::multiseq_partition (
 _RanSeqs __begin_seqs,
 _RanSeqs __end_seqs,
 _RankType __rank,
 _RankIterator __begin_offsets,
 _Compare __comp = std::less< typename std::iterator_traits<typename _RanSeqs>::value_type:: first_type>::value_type>())
```

Splits several sorted sequences at a certain global `__rank`, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the `__left` side will be chosen from sequences with smaller number.

#### Parameters

|                              |                                                                                                                                                                                                                                                              |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code>    | Begin of the sequence of iterator pairs.                                                                                                                                                                                                                     |
| <code>__end_seqs</code>      | End of the sequence of iterator pairs.                                                                                                                                                                                                                       |
| <code>__rank</code>          | The global rank to partition at.                                                                                                                                                                                                                             |
| <code>__begin_offsets</code> | A random-access <code>__sequence</code> <code>__begin</code> where the <code>__result</code> will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective <code>__sequence</code> . |
| <code>__comp</code>          | The ordering functor, defaults to <code>std::less&lt;_Tp&gt;</code> .                                                                                                                                                                                        |

Definition at line 122 of file `multiseq_selection.h`.

References `_GLIBCXX_CALL`, and `std::distance()`.

### 3.6.4.63 multiseq\_selection()

```
template<typename _Tp , typename _RanSeqs , typename _RankType , typename _Compare >
_Tp __gnu_parallel::multiseq_selection (
```



```

_RanSeqs __begin_seqs,
_RanSeqs __end_seqs,
_RankType __rank,
_RankType & __offset,
_Compare __comp = std::less<_Tp>())

```

Selects the element at a certain global `__rank` from several sorted sequences.

The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

#### Parameters

|                           |                                                                                                                                                            |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__begin_seqs</code> | Begin of the sequence of iterator pairs.                                                                                                                   |
| <code>__end_seqs</code>   | End of the sequence of iterator pairs.                                                                                                                     |
| <code>__rank</code>       | The global rank to partition at.                                                                                                                           |
| <code>__offset</code>     | The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0. |
| <code>__comp</code>       | The ordering functor, defaults to <code>std::less</code> .                                                                                                 |

Definition at line 388 of file `multiseq_selection.h`.

References `__rd_log2()`, `__round_up_to_pow2()`, `std::__sample()`, `_GLIBCXX_CALL`, `std::distance()`, and `std::max()`.

#### 3.6.4.64 `multiway_merge()`

```

template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _
_Compare >
_RAITerOut __gnu_parallel::multiway_merge (
 _RAIterPairIterator __seqs_begin,
 _RAIterPairIterator __seqs_end,
 _RAIterOut __target,
 _DifferenceTp __length,
 _Compare __comp,
 __gnu_parallel::sequential_tag)

```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __j < 10; ++__j)
 sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

#### See also

`stable_multiway_merge`

#### Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

#### Postcondition

`[__target, return __value)` contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

#### Template Parameters

|                                  |                                                            |
|----------------------------------|------------------------------------------------------------|
| <code>_RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>_RAIterOut</code>          | iterator over target sequence                              |
| <code>_DifferenceTp</code>       | difference type for the sequence                           |
| <code>_Compare</code>            | strict weak ordering type to compare elements in sequences |

## Parameters

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | __begin of sequence __sequence                                                  |
| <code>__seqs_end</code>   | _M_end of sequence __sequence                                                   |
| <code>__target</code>     | target sequence to merge to.                                                    |
| <code>__comp</code>       | strict weak ordering to use for element comparison.                             |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

## Returns

\_M\_end iterator of output sequence

Definition at line 1418 of file multiway\_merge.h.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

## 3.6.4.65 multiway\_merge\_3\_variant()

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator , typename
_RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)
```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

## Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 241 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

**3.6.4.66 multiway\_merge\_4\_variant()**

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator , typename
_RAIter3 , typename _DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)
```

Highly efficient 4-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 360 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

## 3.6.4.67 multiway\_merge\_exact\_splitting()

```
template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
void __gnu_parallel::multiway_merge_exact_splitting (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _DifferenceType __length,
 _DifferenceType __total_length,
 _Compare __comp,
 std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Exact splitting for parallel multiway-merge routine.

None of the passed sequences may be empty.

Definition at line 1120 of file multiway\_merge.h.

Referenced by \_\_parallel\_merge\_advance().

## 3.6.4.68 multiway\_merge\_loser\_tree()

```
template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp ,
typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _DifferenceTp __length,
 _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.

This merging variant uses a LoserTree class as selected by \_LT.

Stability is selected through the used LoserTree class \_LT.

At least one non-empty sequence is required.

## Parameters

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 491 of file multiway\_merge.h.

References `_GLIBCXX_CALL`, and `_GLIBCXX_PARALLEL_LENGTH`.

**3.6.4.69 multiway\_merge\_loser\_tree\_sentinel()**

```
template<typename _UnguardedLoserTree , typename _RAIterIterator , typename _RAIter3 , typename ↵
_DifferenceTp , typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

**Template Parameters**

|                                  |                                                      |
|----------------------------------|------------------------------------------------------|
| <code>_UnguardedLoserTree</code> | Loser Tree variant to use for the unguarded merging. |
|----------------------------------|------------------------------------------------------|

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 662 of file multiway\_merge.h.

References `_GLIBCXX_CALL`.

## 3.6.4.70 multiway\_merge\_loser\_tree\_unguarded()

```
template<typename _LT , typename _RAIterIterator , typename _RAIter3 , typename _DifferenceTp ,
typename _Compare >
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
 _DifferenceTp __length,
 _Compare __comp)
```

Multi-way merging procedure for a high branching factor, unguarded case.

Merging is done using the LoserTree class `_LT`.

Stability is selected by the used LoserTrees.

**Precondition**

No input will run out of elements during the merge.

**Parameters**

|                           |                                                                                  |
|---------------------------|----------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                  |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                    |
| <code>__target</code>     | Begin iterator of output sequence.                                               |
| <code>__comp</code>       | Comparator.                                                                      |
| <code>__length</code>     | Maximum length to merge, less equal than the total number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 574 of file `multiway_merge.h`.

References `_GLIBCXX_CALL`.

## 3.6.4.71 multiway\_merge\_sampling\_splitting()

```
template<bool __stable, typename _RAIterIterator , typename _Compare , typename _DifferenceType >
void __gnu_parallel::multiway_merge_sampling_splitting (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _DifferenceType __length,
 _DifferenceType __total_length,
```

```
 _Compare __comp,
 std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.

Definition at line 1035 of file multiway\_merge.h.

References `_GLIBCXX_PARALLEL_LENGTH`, `__gnu_parallel::_Settings::get()`, and `__gnu_parallel::_Settings::merge_oversampling`.

#### 3.6.4.72 multiway\_merge\_sentinels()

```
template<typename _RAIterPairIterator , typename _RAIterOut , typename _DifferenceTp , typename _
_Compare >
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
 _RAIterPairIterator __seqs_begin,
 _RAIterPairIterator __seqs_end,
 _RAIterOut __target,
 _DifferenceTp __length,
 _Compare __comp,
 __gnu_parallel::sequential_tag)
```

Multiway Merge Frontend.

Merge the sequences specified by `seqs_begin` and `__seqs_end` into `__target`. `__seqs_begin` and `__seqs_end` must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the `_M_end` of the same sequence in their second entry.

Ties are broken arbitrarily. See `stable_multiway_merge` for a variant that breaks ties by sequence number but is slower.

The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.

The output sequence has to provide enough space for all elements that are written to it.

This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the `_M_end` iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.

Example:



```

int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
 for (int __j = 0; __i < 11; ++__j)
 sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
 { seqs.push(std::make_pair<int*>(sequences[__i],
 sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);

```

**Precondition**

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each `__i`, `__seqs_begin[__i].second` must be the end marker of the sequence, but also reference the one more `__sentinel` element.

**Postcondition**

`[__target, return __value)` contains merged `__elements` from the input sequences.  
`return __value - __target = min(__length, number of elements in all sequences).`

**See also**

`stable_multiway_merge_sentinels`

**Template Parameters**

|                                  |                                                            |
|----------------------------------|------------------------------------------------------------|
| <code>_RAIterPairIterator</code> | iterator over sequence of pairs of iterators               |
| <code>_RAIterOut</code>          | iterator over target sequence                              |
| <code>_DifferenceTp</code>       | difference type for the sequence                           |
| <code>_Compare</code>            | strict weak ordering type to compare elements in sequences |

**Parameters**

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | <code>__begin</code> of sequence <code>__sequence</code>                        |
| <code>__seqs_end</code>   | <code>_M_end</code> of sequence <code>__sequence</code>                         |
| <code>__target</code>     | target sequence to merge to.                                                    |
| <code>__comp</code>       | strict weak ordering to use for element comparison.                             |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

**Returns**

`_M_end` iterator of output sequence

Definition at line 1782 of file `multiway_merge.h`.

References `__sequential_multiway_merge()`, and `_GLIBCXX_CALL`.

**3.6.4.73 parallel\_multiway\_merge()**

```
template<bool __stable, bool __sentinels, typename _RAIterIterator , typename _RAIter3 , typename
_DifferenceTp , typename _Splitter , typename _Compare >
_RAIter3 __gnu_parallel::parallel_multiway_merge (
 _RAIterIterator __seqs_begin,
 _RAIterIterator __seqs_end,
 _RAIter3 __target,
 _Splitter __splitter,
 _DifferenceTp __length,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.

The `_GLIBCXX_PARALLEL_DECISION` is based on the branching factor and runtime settings.

Must not be called if the number of sequences is 1.

**Template Parameters**

|                         |                                                                        |
|-------------------------|------------------------------------------------------------------------|
| <code>_Splitter</code>  | functor to split input (either <code>__exact</code> or sampling based) |
| <code>__stable</code>   | Stable merging incurs a performance penalty.                           |
| <code>__sentinel</code> | Ignored.                                                               |

**Parameters**

|                           |                                                                                 |
|---------------------------|---------------------------------------------------------------------------------|
| <code>__seqs_begin</code> | Begin iterator of iterator pair input sequence.                                 |
| <code>__seqs_end</code>   | End iterator of iterator pair input sequence.                                   |
| <code>__target</code>     | Begin iterator of output sequence.                                              |
| <code>__comp</code>       | Comparator.                                                                     |
| <code>__length</code>     | Maximum length to merge, possibly larger than the number of elements available. |

**Returns**

End iterator of output sequence.

Definition at line 1225 of file `multiway_merge.h`.

Referenced by `__parallel_merge_advance()`.

## 3.6.4.74 parallel\_sort\_mwms()

```
template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
void __gnu_parallel::parallel_sort_mwms (
 _RAIter __begin,
 _RAIter __end,
 _Compare __comp,
 _ThreadIndex __num_threads)
```

PMWMS main call.

## Parameters

|                            |                             |
|----------------------------|-----------------------------|
| <code>__begin</code>       | Begin iterator of sequence. |
| <code>__end</code>         | End iterator of sequence.   |
| <code>__comp</code>        | Comparator.                 |
| <code>__num_threads</code> | Number of threads to use.   |

Definition at line 395 of file multiway\_mergesort.h.

References `_GLIBCXX_CALL`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_num_threads`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_offsets`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_pieces`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_samples`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_source`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_starts`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_temporary`, `__gnu_parallel::Settings::get()`, and `__gnu_parallel::Settings::sort_mwms_oversampling`.

## 3.6.4.75 parallel\_sort\_mwms\_pu()

```
template<bool __stable, bool __exact, typename _RAIter , typename _Compare >
void __gnu_parallel::parallel_sort_mwms_pu (
 _PMWMSortingData< _RAIter > * __sd,
 _Compare & __comp)
```

PMWMS code executed by each thread.

## Parameters

|                     |                            |
|---------------------|----------------------------|
| <code>__sd</code>   | Pointer to algorithm data. |
| <code>__comp</code> | Comparator.                |

Definition at line 308 of file multiway\_mergesort.h.

References `__gnu_parallel::PMWMSortingData< _RAIter >::M_num_threads`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_source`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_starts`, `__gnu_parallel::PMWMSortingData< _RAIter >::M_temporary`, `__gnu_parallel::Settings::get()`, `__gnu_parallel::Settings::sort_mwms_oversampling`, and `std::uninitialized_copy()`.

### 3.6.5 Variable Documentation

#### 3.6.5.1 `_CASable_bits`

```
const int __gnu_parallel::_CASable_bits [static]
```

Number of bits of `_CASable`.

Definition at line 130 of file `types.h`.

Referenced by `__decode2()`, and `__encode2()`.

#### 3.6.5.2 `_CASable_mask`

```
const _CASable __gnu_parallel::_CASable_mask [static]
```

`_CASable` with the right half of bits set to 1.

Definition at line 133 of file `types.h`.

Referenced by `__decode2()`.

## 3.7 `__gnu_pbds` Namespace Reference

### Classes

- struct [associative\\_tag](#)
- class [basic\\_branch](#)
- struct [basic\\_branch\\_tag](#)
- class [basic\\_hash\\_table](#)
- struct [basic\\_hash\\_tag](#)
- struct [basic\\_invalidation\\_guarantee](#)
- struct [binary\\_heap\\_tag](#)
- struct [binomial\\_heap\\_tag](#)
- class [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#)
- class [cc\\_hash\\_table](#)
- struct [cc\\_hash\\_tag](#)
- struct [container\\_error](#)
- struct [container\\_tag](#)
- struct [container\\_traits](#)
- struct [container\\_traits\\_base](#)
- struct [container\\_traits\\_base< binary\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< binomial\\_heap\\_tag >](#)
- struct [container\\_traits\\_base< cc\\_hash\\_tag >](#)
- struct [container\\_traits\\_base< gp\\_hash\\_tag >](#)

- struct `container_traits_base< list_update_tag >`
- struct `container_traits_base< ov_tree_tag >`
- struct `container_traits_base< pairing_heap_tag >`
- struct `container_traits_base< pat_trie_tag >`
- struct `container_traits_base< rb_tree_tag >`
- struct `container_traits_base< rc_binomial_heap_tag >`
- struct `container_traits_base< splay_tree_tag >`
- struct `container_traits_base< thin_heap_tag >`
- class `direct_mask_range_hashing`
- class `direct_mod_range_hashing`
- class `gp_hash_table`
- struct `gp_hash_tag`
- class `hash_exponential_size_policy`
- class `hash_load_check_resize_trigger`
- class `hash_prime_size_policy`
- class `hash_standard_resize_policy`
- struct `insert_error`
- struct `join_error`
- class `linear_probe_fn`
- class `list_update`
- struct `list_update_tag`
- class `lu_counter_policy`
- class `lu_move_to_front_policy`
- struct `null_node_update`
- struct `null_type`
- struct `ov_tree_tag`
- struct `pairing_heap_tag`
- struct `pat_trie_tag`
- struct `point_invalidation_guarantee`
- class `priority_queue`
- struct `priority_queue_tag`
- class `quadratic_probe_fn`
- struct `range_invalidation_guarantee`
- struct `rb_tree_tag`
- struct `rc_binomial_heap_tag`
- struct `resize_error`
- class `sample_probe_fn`
- class `sample_range_hashing`
- class `sample_ranged_hash_fn`
- class `sample_ranged_probe_fn`
- class `sample_resize_policy`
- class `sample_resize_trigger`
- class `sample_size_policy`
- class `sample_tree_node_update`
- struct `sample_trie_access_traits`
- class `sample_trie_node_update`
- struct `sample_update_policy`
- struct `sequence_tag`
- struct `splay_tree_tag`
- struct `string_tag`
- struct `thin_heap_tag`

- class `tree`
- class `tree_order_statistics_node_update`
- struct `tree_tag`
- class `trie`
- class `trie_order_statistics_node_update`
- class `trie_prefix_search_node_update`
- struct `trie_string_access_traits`
- struct `trie_tag`
- struct `trivial_iterator_tag`

#### Typedefs

- typedef void `trivial_iterator_difference_type`

#### Functions

- void `__throw_container_error ()`
- void `__throw_insert_error ()`
- void `__throw_join_error ()`
- void `__throw_resize_error ()`

### 3.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

## 3.8 `__gnu_sequential` Namespace Reference

### 3.8.1 Detailed Description

GNU sequential classes for public use.

## 3.9 `abi` Namespace Reference

### 3.9.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to `__cxxabiv1`, but user programs should use the alias `'abi'`.

A brief overview of an ABI is given in the `libstdc++` FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at [http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5\\_8](http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8)).

GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at <http://www.codesourcery.com/cxx-abi/> along with the current specification.

For users of GCC greater than or equal to 3.x, entry points are available in `<cxxabi.h>`, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

## 3.10 std Namespace Reference

### Namespaces

- [\\_\\_debug](#)
- [\\_\\_detail](#)
- [\\_\\_parallel](#)
- [chrono](#)
- [decimal](#)
- [experimental](#)
- [placeholders](#)
- [regex\\_constants](#)
- [rel\\_ops](#)
- [this\\_thread](#)
- [tr1](#)
- [tr2](#)

### Classes

- [struct \\_\\_add\\_pointer\\_helper](#)
- [struct \\_\\_allocated\\_ptr](#)
- [struct \\_\\_atomic\\_base](#)
- [struct \\_\\_atomic\\_base< \\_PTp \\* >](#)
- [struct \\_\\_atomic\\_flag\\_base](#)
- [class \\_\\_basic\\_future](#)
- [class \\_\\_codecvt\\_abstract\\_base](#)
- [class \\_\\_ctype\\_abstract\\_base](#)
- [struct \\_\\_detector](#)
- [struct \\_\\_detector< \\_Default, \\_\\_void\\_t< \\_Op< \\_Args... > >, \\_Op, \\_Args... >](#)
- [struct \\_\\_future\\_base](#)
- [struct \\_\\_is\\_location\\_invariant](#)
- [struct \\_\\_is\\_nullptr\\_t](#)
- [struct \\_\\_is\\_tuple\\_like\\_impl< std::pair< \\_T1, \\_T2 > >](#)
- [struct \\_\\_numeric\\_limits\\_base](#)
- [struct \\_Base\\_bitset](#)
- [struct \\_Base\\_bitset< 0 >](#)
- [struct \\_Base\\_bitset< 1 >](#)
- [struct \\_Bind](#)
- [struct \\_Bind\\_result](#)
- [class \\_Deque\\_base](#)
- [struct \\_Deque\\_iterator](#)
- [struct \\_Enable\\_copy\\_move](#)
- [struct \\_Enable\\_default\\_constructor](#)
- [struct \\_Enable\\_destructor](#)
- [struct \\_Enable\\_special\\_members](#)
- [class \\_Function\\_base](#)
- [struct \\_Fwd\\_list\\_base](#)
- [struct \\_Fwd\\_list\\_const\\_iterator](#)
- [struct \\_Fwd\\_list\\_iterator](#)
- [struct \\_Fwd\\_list\\_node](#)

- struct [\\_Fwd\\_list\\_node\\_base](#)
- class [\\_Hashtable](#)
- class [\\_List\\_base](#)
- struct [\\_List\\_const\\_iterator](#)
- struct [\\_List\\_iterator](#)
- struct [\\_List\\_node](#)
- class [\\_Mu](#)
- class [\\_Mu< \\_Arg, false, false >](#)
- class [\\_Mu< \\_Arg, false, true >](#)
- class [\\_Mu< \\_Arg, true, false >](#)
- class [\\_Mu< reference\\_wrapper< \\_Tp >, false, false >](#)
- class [\\_Not\\_fn](#)
- struct [\\_Placeholder](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, false >](#)
- struct [\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, true >](#)
- class [\\_Temporary\\_buffer](#)
- struct [\\_Tuple\\_impl](#)
- struct [\\_Tuple\\_impl< \\_Idx, \\_Head, \\_Tail... >](#)
- struct [\\_Vector\\_base](#)
- struct [add\\_const](#)
- struct [add\\_cv](#)
- struct [add\\_lvalue\\_reference](#)
- struct [add\\_rvalue\\_reference](#)
- struct [add\\_volatile](#)
- struct [adopt\\_lock\\_t](#)
- struct [aligned\\_storage](#)
- struct [aligned\\_union](#)
- struct [alignment\\_of](#)
- class [allocator](#)
- class [allocator< void >](#)
- struct [allocator\\_arg\\_t](#)
- struct [allocator\\_traits](#)
- struct [allocator\\_traits< allocator< \\_Tp > >](#)
- struct [array](#)
- struct [atomic](#)
- struct [atomic< \\_Tp \\* >](#)
- struct [atomic< bool >](#)
- struct [atomic< char >](#)
- struct [atomic< char16\\_t >](#)
- struct [atomic< char32\\_t >](#)
- struct [atomic< int >](#)
- struct [atomic< long >](#)
- struct [atomic< long long >](#)
- struct [atomic< short >](#)
- struct [atomic< signed char >](#)
- struct [atomic< unsigned char >](#)
- struct [atomic< unsigned int >](#)
- struct [atomic< unsigned long >](#)
- struct [atomic< unsigned long long >](#)
- struct [atomic< unsigned short >](#)
- struct [atomic< wchar\\_t >](#)



- struct [atomic\\_flag](#)
- class [auto\\_ptr](#)
- struct [auto\\_ptr\\_ref](#)
- class [back\\_insert\\_iterator](#)
- class [bad\\_alloc](#)
- class [bad\\_cast](#)
- class [bad\\_exception](#)
- class [bad\\_function\\_call](#)
- class [bad\\_typeid](#)
- class [bad\\_weak\\_ptr](#)
- class [basic\\_filebuf](#)
- class [basic\\_fstream](#)
- class [basic\\_ifstream](#)
- class [basic\\_ios](#)
- class [basic\\_iostream](#)
- class [basic\\_istream](#)
- class [basic\\_istreambuf](#)
- class [basic\\_ofstream](#)
- class [basic\\_ostream](#)
- class [basic\\_ostringstream](#)
- class [basic\\_regex](#)
- class [basic\\_streambuf](#)
- class [basic\\_string](#)
- class [basic\\_stringbuf](#)
- class [basic\\_stringstream](#)
- class [bernoulli\\_distribution](#)
- struct [bidirectional\\_iterator\\_tag](#)
- struct [binary\\_function](#)
- class [binary\\_negate](#)
- class [binder1st](#)
- class [binder2nd](#)
- class [binomial\\_distribution](#)
- class [bitset](#)
- class [cauchy\\_distribution](#)
- struct [char\\_traits](#)
- struct [char\\_traits< \\_\\_gnu\\_cxx::character< \\_Value, \\_Int, \\_St > >](#)
- struct [char\\_traits< char >](#)
- struct [char\\_traits< wchar\\_t >](#)
- class [chi\\_squared\\_distribution](#)
- class [codecvt](#)
- class [codecvt< \\_InternT, \\_ExternT, encoding\\_state >](#)
- class [codecvt< char, char, mbstate\\_t >](#)
- class [codecvt< char16\\_t, char, mbstate\\_t >](#)
- class [codecvt< char32\\_t, char, mbstate\\_t >](#)
- class [codecvt< wchar\\_t, char, mbstate\\_t >](#)
- class [codecvt\\_base](#)
- class [codecvt\\_byname](#)
- class [collate](#)
- class [collate\\_byname](#)
- struct [common\\_type](#)
- struct [common\\_type< chrono::duration< \\_Rep, \\_Period > >](#)

- struct `common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >`
- struct `common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >`
- struct `common_type< chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >`
- struct `common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >`
- struct `complex`
- struct `complex< double >`
- struct `complex< float >`
- struct `complex< long double >`
- class `condition_variable`
- struct `conditional`
- class `const_mem_fun1_ref_t`
- class `const_mem_fun1_t`
- class `const_mem_fun_ref_t`
- class `const_mem_fun_t`
- class `ctype`
- class `ctype< char >`
- class `ctype< wchar_t >`
- struct `ctype_base`
- class `ctype_byname`
- class `ctype_byname< char >`
- class `decay`
- struct `default_delete`
- struct `default_delete< _Tp[]>`
- struct `defer_lock_t`
- class `deque`
- class `discard_block_engine`
- class `discrete_distribution`
- struct `divides`
- struct `divides< void >`
- class `domain_error`
- struct `enable_if`
- class `enable_shared_from_this`
- struct `equal_to`
- struct `equal_to< void >`
- struct `error_code`
- struct `error_condition`
- class `exception`
- class `exponential_distribution`
- struct `extent`
- class `extreme_value_distribution`
- class `fisher_f_distribution`
- struct `forward_iterator_tag`
- class `forward_list`
- class `fpos`
- struct `from_chars_result`
- class `front_insert_iterator`
- class `function< _Res(_ArgTypes...)>`
- class `future`
- class `future< _Res & >`
- class `future< void >`

- class `future_error`
- class `gamma_distribution`
- class `geometric_distribution`
- struct `greater`
- struct `greater< void >`
- struct `greater_equal`
- struct `greater_equal< void >`
- class `gslice`
- class `gslice_array`
- struct `has_virtual_destructor`
- struct `hash`
- struct `hash< __debug::bitset< _Nb > >`
- struct `hash< __debug::vector< bool, _Alloc > >`
- struct `hash< __gnu_cxx::__u16vstring >`
- struct `hash< __gnu_cxx::__u32vstring >`
- struct `hash< __gnu_cxx::__vstring >`
- struct `hash< __gnu_cxx::__wvstring >`
- struct `hash< __gnu_cxx::throw_value_limit >`
- struct `hash< __gnu_cxx::throw_value_random >`
- struct `hash< __shared_ptr< _Tp, _Lp > >`
- struct `hash< _Tp * >`
- struct `hash< bool >`
- struct `hash< char >`
- struct `hash< char16_t >`
- struct `hash< char32_t >`
- struct `hash< double >`
- struct `hash< error_code >`
- struct `hash< experimental::optional< _Tp > >`
- struct `hash< experimental::shared_ptr< _Tp > >`
- struct `hash< float >`
- struct `hash< int >`
- struct `hash< long >`
- struct `hash< long double >`
- struct `hash< long long >`
- struct `hash< shared_ptr< _Tp > >`
- struct `hash< short >`
- struct `hash< signed char >`
- struct `hash< string >`
- struct `hash< thread::id >`
- struct `hash< type_index >`
- struct `hash< u16string >`
- struct `hash< u32string >`
- struct `hash< unique_ptr< _Tp, _Dp > >`
- struct `hash< unsigned char >`
- struct `hash< unsigned int >`
- struct `hash< unsigned long >`
- struct `hash< unsigned long long >`
- struct `hash< unsigned short >`
- struct `hash< wchar_t >`
- struct `hash< wstring >`
- struct `hash< ::bitset< _Nb > >`

- struct [hash<::vector< bool, \\_Alloc > >](#)
- class [independent\\_bits\\_engine](#)
- class [indirect\\_array](#)
- class [initializer\\_list](#)
- struct [input\\_iterator\\_tag](#)
- class [insert\\_iterator](#)
- struct [integer\\_sequence](#)
- struct [integral\\_constant](#)
- class [invalid\\_argument](#)
- class [ios\\_base](#)
- struct [is\\_abstract](#)
- struct [is\\_arithmetic](#)
- struct [is\\_array](#)
- struct [is\\_assignable](#)
- struct [is\\_base\\_of](#)
- struct [is\\_bind\\_expression](#)
- struct [is\\_bind\\_expression< \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< const volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_bind\\_expression< volatile \\_Bind< \\_Signature > >](#)
- struct [is\\_bind\\_expression< volatile \\_Bind\\_result< \\_Result, \\_Signature > >](#)
- struct [is\\_class](#)
- struct [is\\_compound](#)
- struct [is\\_const](#)
- struct [is\\_constructible](#)
- struct [is\\_convertible](#)
- struct [is\\_copy\\_assignable](#)
- struct [is\\_copy\\_constructible](#)
- struct [is\\_default\\_constructible](#)
- struct [is\\_destructible](#)
- struct [is\\_empty](#)
- struct [is\\_enum](#)
- struct [is\\_error\\_code\\_enum](#)
- struct [is\\_error\\_code\\_enum< future\\_errc >](#)
- struct [is\\_error\\_condition\\_enum](#)
- struct [is\\_final](#)
- struct [is\\_floating\\_point](#)
- struct [is\\_function](#)
- struct [is\\_fundamental](#)
- struct [is\\_integral](#)
- struct [is\\_literal\\_type](#)
- struct [is\\_lvalue\\_reference](#)
- struct [is\\_member\\_function\\_pointer](#)
- struct [is\\_member\\_object\\_pointer](#)
- struct [is\\_member\\_pointer](#)
- struct [is\\_move\\_assignable](#)
- struct [is\\_move\\_constructible](#)
- struct [is\\_nothrow\\_assignable](#)

- struct [is\\_nothrow\\_constructible](#)
- struct [is\\_nothrow\\_copy\\_assignable](#)
- struct [is\\_nothrow\\_copy\\_constructible](#)
- struct [is\\_nothrow\\_default\\_constructible](#)
- struct [is\\_nothrow\\_destructible](#)
- struct [is\\_nothrow\\_move\\_assignable](#)
- struct [is\\_nothrow\\_move\\_constructible](#)
- struct [is\\_nothrow\\_swappable](#)
- struct [is\\_nothrow\\_swappable\\_with](#)
- struct [is\\_null\\_pointer](#)
- struct [is\\_object](#)
- struct [is\\_placeholder](#)
- struct [is\\_placeholder< \\_Placeholder< \\_Num > >](#)
- struct [is\\_pod](#)
- struct [is\\_pointer](#)
- struct [is\\_polymorphic](#)
- struct [is\\_reference](#)
- struct [is\\_rvalue\\_reference](#)
- struct [is\\_same](#)
- struct [is\\_scalar](#)
- struct [is\\_standard\\_layout](#)
- struct [is\\_swappable](#)
- struct [is\\_swappable\\_with](#)
- struct [is\\_trivial](#)
- struct [is\\_trivially\\_assignable](#)
- struct [is\\_trivially\\_constructible](#)
- struct [is\\_trivially\\_copy\\_assignable](#)
- struct [is\\_trivially\\_copy\\_constructible](#)
- struct [is\\_trivially\\_default\\_constructible](#)
- struct [is\\_trivially\\_destructible](#)
- struct [is\\_trivially\\_move\\_assignable](#)
- struct [is\\_trivially\\_move\\_constructible](#)
- struct [is\\_union](#)
- struct [is\\_void](#)
- struct [is\\_volatile](#)
- class [istream\\_iterator](#)
- class [istreambuf\\_iterator](#)
- struct [iterator](#)
- struct [iterator\\_traits](#)
- struct [iterator\\_traits< \\_Tp \\* >](#)
- struct [iterator\\_traits< const \\_Tp \\* >](#)
- class [length\\_error](#)
- struct [less](#)
- struct [less< void >](#)
- struct [less\\_equal](#)
- struct [less\\_equal< void >](#)
- class [linear\\_congruential\\_engine](#)
- class [list](#)
- class [locale](#)
- class [lock\\_guard](#)
- class [logic\\_error](#)

- struct [logical\\_and](#)
- struct [logical\\_and< void >](#)
- struct [logical\\_not](#)
- struct [logical\\_not< void >](#)
- struct [logical\\_or](#)
- struct [logical\\_or< void >](#)
- class [lognormal\\_distribution](#)
- struct [make\\_signed](#)
- struct [make\\_unsigned](#)
- class [map](#)
- class [mask\\_array](#)
- class [match\\_results](#)
- class [mem\\_fun1\\_ref\\_t](#)
- class [mem\\_fun1\\_t](#)
- class [mem\\_fun\\_ref\\_t](#)
- class [mem\\_fun\\_t](#)
- class [mersenne\\_twister\\_engine](#)
- class [messages](#)
- struct [messages\\_base](#)
- class [messages\\_byname](#)
- struct [minus](#)
- struct [minus< void >](#)
- struct [modulus](#)
- struct [modulus< void >](#)
- class [money\\_base](#)
- class [money\\_get](#)
- class [money\\_put](#)
- class [moneypunct](#)
- class [moneypunct\\_byname](#)
- class [move\\_iterator](#)
- class [multimap](#)
- struct [multiplies](#)
- struct [multiplies< void >](#)
- class [multiset](#)
- class [mutex](#)
- struct [negate](#)
- struct [negate< void >](#)
- class [negative\\_binomial\\_distribution](#)
- class [nested\\_exception](#)
- class [normal\\_distribution](#)
- struct [not\\_equal\\_to](#)
- struct [not\\_equal\\_to< void >](#)
- class [num\\_get](#)
- class [num\\_put](#)
- struct [numeric\\_limits](#)
- struct [numeric\\_limits< bool >](#)
- struct [numeric\\_limits< char >](#)
- struct [numeric\\_limits< char16\\_t >](#)
- struct [numeric\\_limits< char32\\_t >](#)
- struct [numeric\\_limits< double >](#)
- struct [numeric\\_limits< float >](#)

- struct `numeric_limits< int >`
- struct `numeric_limits< long >`
- struct `numeric_limits< long double >`
- struct `numeric_limits< long long >`
- struct `numeric_limits< short >`
- struct `numeric_limits< signed char >`
- struct `numeric_limits< unsigned char >`
- struct `numeric_limits< unsigned int >`
- struct `numeric_limits< unsigned long >`
- struct `numeric_limits< unsigned long long >`
- struct `numeric_limits< unsigned short >`
- struct `numeric_limits< wchar_t >`
- class `numpunct`
- class `numpunct_byname`
- struct `once_flag`
- class `ostream_iterator`
- class `ostreambuf_iterator`
- class `out_of_range`
- struct `output_iterator_tag`
- class `overflow_error`
- struct `owner_less`
- struct `owner_less< shared_ptr< _Tp > >`
- struct `owner_less< void >`
- struct `owner_less< weak_ptr< _Tp > >`
- class `packaged_task< _Res(_ArgTypes...)>`
- struct `pair`
- class `piecewise_constant_distribution`
- struct `piecewise_construct_t`
- class `piecewise_linear_distribution`
- struct `plus`
- class `pointer_to_binary_function`
- class `pointer_to_unary_function`
- struct `pointer_traits`
- struct `pointer_traits< _Tp * >`
- class `poisson_distribution`
- class `priority_queue`
- class `promise`
- class `promise< _Res & >`
- class `promise< void >`
- class `queue`
- struct `random_access_iterator_tag`
- class `random_device`
- class `range_error`
- struct `rank`
- struct `ratio`
- struct `ratio_equal`
- struct `ratio_greater`
- struct `ratio_greater_equal`
- struct `ratio_less`
- struct `ratio_less_equal`
- struct `ratio_not_equal`

- class [raw\\_storage\\_iterator](#)
- class [recursive\\_mutex](#)
- class [recursive\\_timed\\_mutex](#)
- class [reference\\_wrapper](#)
- class [regex\\_error](#)
- class [regex\\_iterator](#)
- class [regex\\_token\\_iterator](#)
- class [regex\\_traits](#)
- struct [remove\\_all\\_extents](#)
- struct [remove\\_const](#)
- struct [remove\\_cv](#)
- struct [remove\\_extent](#)
- struct [remove\\_pointer](#)
- struct [remove\\_reference](#)
- struct [remove\\_volatile](#)
- class [result\\_of](#)
- class [reverse\\_iterator](#)
- class [runtime\\_error](#)
- class [scoped\\_allocator\\_adaptor](#)
- class [seed\\_seq](#)
- class [set](#)
- class [shared\\_future](#)
- class [shared\\_future< \\_Res & >](#)
- class [shared\\_future< void >](#)
- class [shared\\_lock](#)
- class [shared\\_ptr](#)
- class [shared\\_timed\\_mutex](#)
- class [shuffle\\_order\\_engine](#)
- class [slice](#)
- class [slice\\_array](#)
- class [stack](#)
- class [student\\_t\\_distribution](#)
- class [sub\\_match](#)
- class [subtract\\_with\\_carry\\_engine](#)
- class [system\\_error](#)
- class [thread](#)
- class [time\\_base](#)
- class [time\\_get](#)
- class [time\\_get\\_byname](#)
- class [time\\_put](#)
- class [time\\_put\\_byname](#)
- class [timed\\_mutex](#)
- struct [to\\_chars\\_result](#)
- struct [try\\_to\\_lock\\_t](#)
- class [tuple](#)
- class [tuple< \\_T1, \\_T2 >](#)
- struct [tuple\\_element](#)
- struct [tuple\\_element< 0, std::pair< \\_Tp1, \\_Tp2 > >](#)
- struct [tuple\\_element< 0, tuple< \\_Head, \\_Tail... > >](#)
- struct [tuple\\_element< 1, std::pair< \\_Tp1, \\_Tp2 > >](#)
- struct [tuple\\_element< \\_\\_i, tuple< \\_Head, \\_Tail... > >](#)



- struct `tuple_element< __i, tuple<> >`
- struct `tuple_element< _Int, ::array< _Tp, _Nm > >`
- struct `tuple_element< _Int, std::__debug::array< _Tp, _Nm > >`
- struct `tuple_size`
- struct `tuple_size< std::__debug::array< _Tp, _Nm > >`
- struct `tuple_size< std::pair< _Tp1, _Tp2 > >`
- struct `tuple_size< tuple< _Elements... > >`
- struct `tuple_size<::array< _Tp, _Nm > >`
- struct `type_index`
- class `type_info`
- struct `unary_function`
- class `unary_negate`
- class `underflow_error`
- struct `underlying_type`
- class `uniform_int_distribution`
- class `uniform_real_distribution`
- class `unique_lock`
- class `unique_ptr`
- class `unique_ptr< _Tp[], _Dp >`
- class `unordered_map`
- class `unordered_multimap`
- class `unordered_multiset`
- class `unordered_set`
- struct `uses_allocator`
- struct `uses_allocator< tuple< _Types... >, _Alloc >`
- class `valarray`
- class `vector`
- class `vector< bool, _Alloc >`
- class `wbuffer_convert`
- class `weak_ptr`
- class `weibull_distribution`
- class `wstring_convert`

## Typedefs

- `template<typename _Alloc, typename _Up >`  
using `__alloc_rebind` = `typename __allocator_traits_base::template __rebind< _Alloc, _Up >::type`
- `template<typename _Tp >`  
using `__allocator_base` = `__gnu_cxx::new_allocator< _Tp >`
- `template<typename _Fn, typename... _Args>`  
using `__async_result_of` = `typename __invoke_result< typename decay< _Fn >::type, typename decay< _↔  
Args >::type... >::type`
- `template<typename _Tp >`  
using `__atomic_diff_t` = `typename atomic< _Tp >::difference_type`
- `typedef unsigned char __atomic_flag_data_type`
- `template<typename _Tp >`  
using `__atomic_val_t` = `typename atomic< _Tp >::value_type`
- `template<bool __v>`  
using `__bool_constant` = `integral_constant< bool, __v >`
- `typedef FILE __c_file`

- `typedef __locale_t __c_locale`
- `typedef __pthread_mutex_t __c_lock`
- `template<typename _Tp, typename _Hash >`  
`using __cache_default = __not_< __and_< __is_fast_hash< _Hash >, __is_nothrow_invocable< const _Hash`  
`&, const _Tp & > >>`
- `template<typename _Fn, typename... _Args>`  
`using __call_is_nothrow = __call_is_nothrow< __invoke_result< _Fn, _Args... >, _Fn, _Args... >`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`using __can_invoke_as_nonvoid = __enable_if_t< __and_< __not_< is_void< _Res > >, is_convertible<`  
`typename __invoke_result< _Callable, _Args... >::type, _Res > >::value, _Res >`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`using __can_invoke_as_void = __enable_if_t< __and_< is_void< _Res >, __is_invocable< _Callable, _Args...`  
`> >::value, _Res >`
- `typedef basic_string< char > __cow_string`
- `template<typename _Tp >`  
`using __decay_and_strip = __strip_reference_wrapper< __decay_t< _Tp > >`
- `template<typename _Tp >`  
`using __decay_t = typename decay< _Tp >::type`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using __detected_or = __detector< _Default, void, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using __detected_or_t = typename __detected_or< _Default, _Op, _Args... >::type`
- `template<typename _Tp >`  
`using __do_is_convertible_to_basic_istream_impl = decltype(__is_convertible_to_basic_istream_↵`  
`test(declval< typename remove_reference< _Tp >::type * >()))`
- `template<typename _Tp >`  
`using __do_is_convertible_to_basic_ostream_impl = decltype(__is_convertible_to_basic_ostream_↵`  
`test(declval< typename remove_reference< _Tp >::type * >()))`
- `template<typename _Tp >`  
`using __empty_not_final = typename conditional< __is_final(_Tp), false_type, __is_empty_non_tuple< _Tp`  
`>::type`
- `template<typename _Tp, typename _Up = typename remove_cv< _Tp >::type, typename = typename enable_if<is_same< _Tp, _Up>↵`  
`::value>::type, size_t = tuple_size< _Tp >::value>`  
`using __enable_if_has_tuple_size = _Tp`
- `template<bool _Cond, typename _Tp = void>`  
`using __enable_if_t = typename enable_if< _Cond, _Tp >::type`
- `template<typename _Tp >`  
`using __get_first_arg_t = typename __get_first_arg< _Tp >::type`
- `template<typename _Cmp, typename _SfinaeType >`  
`using __has_is_transparent_t = typename __has_is_transparent< _Cmp, _SfinaeType >::type`
- `template<typename _ToElementType, typename _FromElementType >`  
`using __is_array_convertible = is_convertible< _FromElementType(*)[], _ToElementType(*)[]>`
- `template<typename _Alloc, typename _Tp >`  
`using __is_erased_or_convertible = __or_< is_convertible< _Alloc, _Tp >, is_same< _Tp, __erased_type`  
`>`
- `template<typename _Tp, typename... _Args>`  
`using __is_nothrow_constructible_impl = __is_nt_constructible_impl< __is_constructible(_Tp, _Args...), _Tp,`  
`_Args... >`
- `template<typename _Tp, typename... _Types>`  
`using __is_one_of = __or_< is_same< _Tp, _Types >... >`
- `template<typename _Tp >`  
`using __is_signed_integer = __is_one_of< __remove_cv_t< _Tp >, signed char, signed short, signed int,`  
`signed long, signed long long >`

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>`  
`using __is_socketlike = __or_< is\_integral< _Tp2 >, is\_enum< _Tp2 > >`
- `template<typename _Tp >`  
`using __is_standard_integer = __or_< is\_signed\_integer< _Tp >, is\_unsigned\_integer< _Tp > >`
- `template<typename _Tp >`  
`using __is_unsigned_integer = is\_one\_of< remove\_cv\_t< _Tp >, unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long >`
- `template<typename _Iter >`  
`using __iterator_category_t = typename iterator\_traits< _Iter >::iterator_category`
- `template<typename _Tp >`  
`using __make_not_void = typename conditional< is\_void< _Tp >::value, __undefined, _Tp >::type`
- `template<typename _Ptr, typename _Tp >`  
`using __ptr_rebind = typename pointer\_traits< _Ptr >::template rebind< _Tp >`
- `template<typename _Tp >`  
`using __remove_cv_t = typename remove\_cv< _Tp >::type`
- `template<typename _Tp >`  
`using __remove_cvref_t = typename remove\_cv< typename remove\_reference< _Tp >::type >::type`
- `template<typename _Tp, typename _Up >`  
`using __replace_first_arg_t = typename \_\_replace\_first\_arg< _Tp, _Up >::type`
- `template<typename _Istream >`  
`using __rvalue_istream_type = typename \_\_is\_convertible\_to\_basic\_istream< _Istream >::__istream_type`
- `template<typename _Ostream >`  
`using __rvalue_ostream_type = typename \_\_is\_convertible\_to\_basic\_ostream< _Ostream >::__ostream_type`
- `typedef basic\_string< char > __sso_string`
- `template<std::size_t _i, typename _Tp >`  
`using __tuple_element_t = typename tuple\_element< _i, _Tp >::type`
- `template<typename _Tp >`  
`using __type_identity_t = typename \_\_type\_identity< _Tp >::type`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __umap_hashtable = Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::Select1st, __Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`  
`using __umap_traits = __detail::Hashtable_traits< _Cache, false, true >`
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>`  
`using __ummap_hashtable = Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::Select1st, __Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`  
`using __ummap_traits = __detail::Hashtable_traits< _Cache, false, false >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __umset_hashtable = Hashtable< _Value, _Value, _Alloc, __detail::Identity, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`
- `template<bool _Cache>`  
`using __umset_traits = __detail::Hashtable_traits< _Cache, true, false >`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`using __uses_alloc_t = __uses_alloc< uses\_allocator< _Tp, _Alloc >::value, _Tp, _Alloc, _Args... >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>`  
`using __uset_hashtable = Hashtable< _Value, _Value, _Alloc, __detail::Identity, _Pred, _Hash, __detail::Mod_range_hashing, __detail::Default_ranged_hash, __detail::Prime_rehash_policy, _Tr >`

- `template<bool _Cache>`  
`using __uset_traits = __detail::__Hashtable_traits< _Cache, true, true >`
- `template<typename... >`  
`using __void_t = void`
- `typedef unsigned long _Bit_type`
- `template<typename... _Cond>`  
`using _Require = __enable_if_t< __and< _Cond... >::value >`
- `template<typename _Alloc >`  
`using _RequireAllocator = typename enable_if< __is_allocator< _Alloc >::value, _Alloc >::type`
- `template<typename _InIter >`  
`using _RequireInputIter = __enable_if_t< is_convertible< __iterator_category_t< _InIter >, input_iterator_tag >::value >`
- `template<typename _Alloc >`  
`using _RequireNotAllocator = typename enable_if<!__is_allocator< _Alloc >::value, _Alloc >::type`
- `template<std::size_t __i, typename _Tuple >`  
`using _Safe_tuple_element_t = typename enable_if<(__i < tuple_size< _Tuple >::value), tuple_element< __i, _Tuple > >::type::type`
- `template<typename _Tp >`  
`using add_const_t = typename add_const< _Tp >::type`
- `template<typename _Tp >`  
`using add_cv_t = typename add_cv< _Tp >::type`
- `template<typename _Tp >`  
`using add_lvalue_reference_t = typename add_lvalue_reference< _Tp >::type`
- `template<typename _Tp >`  
`using add_pointer_t = typename add_pointer< _Tp >::type`
- `template<typename _Tp >`  
`using add_rvalue_reference_t = typename add_rvalue_reference< _Tp >::type`
- `template<typename _Tp >`  
`using add_volatile_t = typename add_volatile< _Tp >::type`
- `template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa< _Len >::__type)>`  
`using aligned_storage_t = typename aligned_storage< _Len, _Align >::type`
- `template<size_t _Len, typename... _Types>`  
`using aligned_union_t = typename aligned_union< _Len, _Types... >::type`
- `typedef atomic< bool > atomic_bool`
- `typedef atomic< char > atomic_char`
- `typedef atomic< char16_t > atomic_char16_t`
- `typedef atomic< char32_t > atomic_char32_t`
- `typedef atomic< int > atomic_int`
- `typedef atomic< int16_t > atomic_int16_t`
- `typedef atomic< int32_t > atomic_int32_t`
- `typedef atomic< int64_t > atomic_int64_t`
- `typedef atomic< int8_t > atomic_int8_t`
- `typedef atomic< int_fast16_t > atomic_int_fast16_t`
- `typedef atomic< int_fast32_t > atomic_int_fast32_t`
- `typedef atomic< int_fast64_t > atomic_int_fast64_t`
- `typedef atomic< int_fast8_t > atomic_int_fast8_t`
- `typedef atomic< int_least16_t > atomic_int_least16_t`
- `typedef atomic< int_least32_t > atomic_int_least32_t`
- `typedef atomic< int_least64_t > atomic_int_least64_t`
- `typedef atomic< int_least8_t > atomic_int_least8_t`
- `typedef atomic< intmax_t > atomic_intmax_t`
- `typedef atomic< intptr_t > atomic_intptr_t`

- typedef [atomic](#)< long long > [atomic\\_llong](#)
- typedef [atomic](#)< long > [atomic\\_long](#)
- typedef [atomic](#)< ptrdiff\_t > [atomic\\_ptrdiff\\_t](#)
- typedef [atomic](#)< signed char > [atomic\\_schar](#)
- typedef [atomic](#)< short > [atomic\\_short](#)
- typedef [atomic](#)< size\_t > [atomic\\_size\\_t](#)
- typedef [atomic](#)< unsigned char > [atomic\\_uchar](#)
- typedef [atomic](#)< unsigned int > [atomic\\_uint](#)
- typedef [atomic](#)< uint16\_t > [atomic\\_uint16\\_t](#)
- typedef [atomic](#)< uint32\_t > [atomic\\_uint32\\_t](#)
- typedef [atomic](#)< uint64\_t > [atomic\\_uint64\\_t](#)
- typedef [atomic](#)< uint8\_t > [atomic\\_uint8\\_t](#)
- typedef [atomic](#)< uint\_fast16\_t > [atomic\\_uint\\_fast16\\_t](#)
- typedef [atomic](#)< uint\_fast32\_t > [atomic\\_uint\\_fast32\\_t](#)
- typedef [atomic](#)< uint\_fast64\_t > [atomic\\_uint\\_fast64\\_t](#)
- typedef [atomic](#)< uint\_fast8\_t > [atomic\\_uint\\_fast8\\_t](#)
- typedef [atomic](#)< uint\_least16\_t > [atomic\\_uint\\_least16\\_t](#)
- typedef [atomic](#)< uint\_least32\_t > [atomic\\_uint\\_least32\\_t](#)
- typedef [atomic](#)< uint\_least64\_t > [atomic\\_uint\\_least64\\_t](#)
- typedef [atomic](#)< uint\_least8\_t > [atomic\\_uint\\_least8\\_t](#)
- typedef [atomic](#)< uintmax\_t > [atomic\\_uintmax\\_t](#)
- typedef [atomic](#)< uintptr\_t > [atomic\\_uintptr\\_t](#)
- typedef [atomic](#)< unsigned long long > [atomic\\_ullong](#)
- typedef [atomic](#)< unsigned long > [atomic\\_ulong](#)
- typedef [atomic](#)< unsigned short > [atomic\\_ushort](#)
- typedef [atomic](#)< wchar\_t > [atomic\\_wchar\\_t](#)
- typedef [ratio](#)< 1, 1000000000000000000 > **atto**
- typedef [ratio](#)< 1, 100 > **centi**
- typedef [match\\_results](#)< const char \* > **cmatch**
- template<typename... \_Tp>  
using [common\\_type\\_t](#) = typename [common\\_type](#)< \_Tp... >::type
- template<bool \_Cond, typename \_Iftrue, typename \_Iffalse >  
using [conditional\\_t](#) = typename [conditional](#)< \_Cond, \_Iftrue, \_Iffalse >::type
- typedef [regex\\_iterator](#)< const char \* > **cregex\_iterator**
- typedef [regex\\_token\\_iterator](#)< const char \* > [cregex\\_token\\_iterator](#)
- typedef [sub\\_match](#)< const char \* > [csub\\_match](#)
- typedef [ratio](#)< 10, 1 > **deca**
- template<typename \_Tp >  
using [decay\\_t](#) = typename [decay](#)< \_Tp >::type
- typedef [ratio](#)< 1, 10 > **deci**
- typedef [minstd\\_rand0](#) **default\_random\_engine**
- template<bool \_Cond, typename \_Tp = void>  
using [enable\\_if\\_t](#) = typename [enable\\_if](#)< \_Cond, \_Tp >::type
- typedef [ratio](#)< 1000000000000000000, 1 > **exa**
- typedef [integral\\_constant](#)< bool, false > [false\\_type](#)
- typedef [ratio](#)< 1, 1000000000000000000 > **femto**
- typedef [basic\\_filebuf](#)< char > [filebuf](#)
- typedef [basic\\_fstream](#)< char > [fstream](#)
- typedef [ratio](#)< 1000000000, 1 > **giga**
- typedef [ratio](#)< 100, 1 > **hecto**
- typedef [basic\\_ifstream](#)< char > [ifstream](#)

---

```

• template<size_t... _Idx>
 using index_sequence = integer_sequence< size_t, _Idx... >
• template<typename... _Types>
 using index_sequence_for = make_index_sequence< sizeof...(_Types)>
• typedef basic_ios< char > ios
• typedef basic_iostream< char > iostream
• typedef basic_istream< char > istream
• typedef basic_ostream< char > ostream
• typedef basic_stringstream< char > istringstream
• typedef ratio< 1000, 1 > kilo
• typedef shuffle_order_engine< minstd_rand0, 256 > knuth_b
• template<size_t _Num>
 using make_index_sequence = make_integer_sequence< size_t, _Num >
• template<typename _Tp, _Tp _Num>
 using make_integer_sequence = integer_sequence< _Tp, __integer_pack(_Num)... >
• template<typename _Tp >
 using make_signed_t = typename make_signed< _Tp >::type
• template<typename _Tp >
 using make_unsigned_t = typename make_unsigned< _Tp >::type
• typedef ratio< 1000000, 1 > mega
• typedef enum std::memory_order memory_order
• typedef ratio< 1, 1000000 > micro
• typedef ratio< 1, 1000 > milli
• typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand
• typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0
• typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7,
 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937
• typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > mt19937_64
• typedef ratio< 1, 1000000000 > nano
• typedef void(* new_handler) ()
• typedef basic_ofstream< char > ofstream
• typedef basic_ostream< char > ostream
• typedef basic_ostringstream< char > ostringstream
• typedef ratio< 1000000000000000, 1 > peta
• typedef ratio< 1, 1000000000000 > pico
• typedef __PTRDIFF_TYPE__ ptrdiff_t
• typedef discard_block_engine< ranlux24_base, 223, 23 > ranlux24
• typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > ranlux24_base
• typedef discard_block_engine< ranlux48_base, 389, 11 > ranlux48
• typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > ranlux48_base
• template<typename _R1, typename _R2 >
 using ratio_add = typename __ratio_add< _R1, _R2 >::type
• template<typename _R1, typename _R2 >
 using ratio_divide = typename __ratio_divide< _R1, _R2 >::type
• template<typename _R1, typename _R2 >
 using ratio_multiply = typename __ratio_multiply< _R1, _R2 >::type
• template<typename _R1, typename _R2 >
 using ratio_subtract = typename __ratio_subtract< _R1, _R2 >::type
• typedef basic_regex< char > regex
• template<typename _Tp >
 using remove_all_extents_t = typename remove_all_extents< _Tp >::type

```

---

- `template<typename _Tp >`  
  using `remove_const_t` = `typename remove_const< _Tp >::type`
- `template<typename _Tp >`  
  using `remove_cv_t` = `typename remove_cv< _Tp >::type`
- `template<typename _Tp >`  
  using `remove_extent_t` = `typename remove_extent< _Tp >::type`
- `template<typename _Tp >`  
  using `remove_pointer_t` = `typename remove_pointer< _Tp >::type`
- `template<typename _Tp >`  
  using `remove_reference_t` = `typename remove_reference< _Tp >::type`
- `template<typename _Tp >`  
  using `remove_volatile_t` = `typename remove_volatile< _Tp >::type`
- `template<typename _Tp >`  
  using `result_of_t` = `typename result_of< _Tp >::type`
- `typedef __SIZE_TYPE__ size_t`
- `typedef match_results< string::const_iterator > smatch`
- `typedef regex_iterator< string::const_iterator > sregex_iterator`
- `typedef regex_token_iterator< string::const_iterator > sregex_token_iterator`
- `typedef sub_match< string::const_iterator > ssb_match`
- `typedef basic_streambuf< char > streambuf`
- `typedef long long streamoff`
- `typedef fpos< mbstate_t > streampos`
- `typedef ptrdiff_t streamsize`
- `typedef basic_string< char > string`
- `typedef basic_stringbuf< char > stringbuf`
- `typedef basic_stringstream< char > stringstream`
- `typedef ratio< 1000000000000, 1 > tera`
- `typedef void(* terminate_handler) ()`
- `typedef integral_constant< bool, true > true_type`
- `template<std::size_t __i, typename _Tp >`  
  using `tuple_element_t` = `typename tuple_element< __i, _Tp >::type`
- `typedef fpos< mbstate_t > u16streampos`
- `typedef basic_string< char16_t > u16string`
- `typedef fpos< mbstate_t > u32streampos`
- `typedef basic_string< char32_t > u32string`
- `template<typename _Tp >`  
  using `underlying_type_t` = `typename underlying_type< _Tp >::type`
- `typedef void(* unexpected_handler) ()`
- `template<typename... >`  
  using `void_t` = `void`
- `typedef match_results< const wchar_t * > wcmatch`
- `typedef regex_iterator< const wchar_t * > wcregex_iterator`
- `typedef regex_token_iterator< const wchar_t * > wcregex_token_iterator`
- `typedef sub_match< const wchar_t * > wsub_match`
- `typedef basic_filebuf< wchar_t > wfilebuf`
- `typedef basic_fstream< wchar_t > wfstream`
- `typedef basic_ifstream< wchar_t > wifstream`
- `typedef basic_ios< wchar_t > wios`
- `typedef basic_iostream< wchar_t > wiostream`
- `typedef basic_istream< wchar_t > wistream`
- `typedef basic_istreambuf_iterator< wchar_t > wistreambuf_iterator`

- typedef `basic_ofstream`< `wchar_t` > `wofstream`
- typedef `basic_ostream`< `wchar_t` > `wostream`
- typedef `basic_ostringstream`< `wchar_t` > `wostringstream`
- typedef `basic_regex`< `wchar_t` > `wregex`
- typedef `match_results`< `wstring::const_iterator` > `wsmatch`
- typedef `regex_iterator`< `wstring::const_iterator` > `wsregex_iterator`
- typedef `regex_token_iterator`< `wstring::const_iterator` > `wsregex_token_iterator`
- typedef `sub_match`< `wstring::const_iterator` > `wssub_match`
- typedef `basic_streambuf`< `wchar_t` > `wstreambuf`
- typedef `fpos`< `mbstate_t` > `wstreampos`
- typedef `basic_string`< `wchar_t` > `wstring`
- typedef `basic_stringbuf`< `wchar_t` > `wstringbuf`
- typedef `basic_stringstream`< `wchar_t` > `wstringstream`

## Enumerations

- enum { `_S_threshold` }
- enum { `_S_chunk_size` }
- enum { `_S_word_bit` }
- enum `__memory_order_modifier` { `__memory_order_mask`, `__memory_order_modifier_mask`, `__memory_order_hle_acquire`, `__memory_order_hle_release` }
- enum `_ios_Fmtflags` { `_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`, `_S_internal`, `_S_left`, `_S_oct`, `_S_right`, `_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`, `_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`, `_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end`, `_S_ios_fmtflags_max`, `_S_ios_fmtflags_min` }
- enum `_ios_iostate` { `_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`, `_S_ios_iostate_end`, `_S_ios_iostate_max`, `_S_ios_iostate_min` }
- enum `_ios_Openmode` { `_S_app`, `_S_ate`, `_S_bin`, `_S_in`, `_S_out`, `_S_trunc`, `_S_ios_openmode_end`, `_S_ios_openmode_max`, `_S_ios_openmode_min` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum `_Manager_operation` { `__get_type_info`, `__get_func_ptr`, `__clone_func_ptr`, `__destroy_func_ptr` }
- enum `_Rb_tree_color` { `_S_red`, `_S_black` }
- enum `chars_format` { `scientific`, `fixed`, `hex`, `general` }
- enum `codecvt_mode` { `consume_header`, `generate_header`, `little_endian` }
- enum `cv_status` { `no_timeout`, `timeout` }
- enum `errc` { `address_family_not_supported`, `address_in_use`, `address_not_available`, `already_connected`, `argument_list_too_long`, `argument_out_of_domain`, `bad_address`, `bad_file_descriptor`, `broken_pipe`, `connection_aborted`, `connection_already_in_progress`, `connection_refused`, `connection_reset`, `cross_device_link`, `destination_address_required`, `device_or_resource_busy`, `directory_not_empty`, `executable_format_error`, `file_exists`, `file_too_large`, `filename_too_long`, `function_not_supported`, `host_unreachable`, `illegal_byte_sequence`, `inappropriate_io_control_operation`, `interrupted`, `invalid_argument`, `invalid_seek`, `io_error`, `is_a_directory`, `message_size`, `network_down`, `network_reset`, `network_unreachable`, `no_buffer_space`, `no_child_process`,



- `no_lock_available`, `no_message`, `no_protocol_option`, `no_space_on_device`,  
`no_such_device_or_address`, `no_such_device`, `no_such_file_or_directory`, `no_such_process`,  
`not_a_directory`, `not_a_socket`, `not_connected`, `not_enough_memory`,  
`operation_in_progress`, `operation_not_permitted`, `operation_not_supported`, `operation_would_block`,  
`permission_denied`, `protocol_not_supported`, `read_only_file_system`, `resource_deadlock_would_occur`,  
`resource_unavailable_try_again`, `result_out_of_range`, `timed_out`, `too_many_files_open_in_system`,  
`too_many_files_open`, `too_many_links`, `too_many_symbolic_link_levels`, `wrong_protocol_type` }
- enum `float_denorm_style` { `denorm_indeterminate`, `denorm_absent`, `denorm_present` }
- enum `float_round_style` {  
`round_indeterminate`, `round_toward_zero`, `round_to_nearest`, `round_toward_infinity`,  
`round_toward_neg_infinity` }
- enum `future_errc` { `future_already_retrieved`, `promise_already_satisfied`, `no_state`, `broken_promise` }
- enum `future_status` { `ready`, `timeout`, `deferred` }
- enum `io_errc` { `stream` }
- enum `launch` { `async`, `deferred` }
- enum `memory_order` {  
`memory_order_relaxed`, `memory_order_consume`, `memory_order_acquire`, `memory_order_release`,  
`memory_order_acq_rel`, `memory_order_seq_cst` }
- enum `pointer_safety` { `relaxed`, `preferred`, `strict` }

## Functions

- template<typename \_CharT >  
`_CharT * __add_grouping` ( `_CharT * __s`, `_CharT __sep`, `const char * __gbeg`, `size_t __gsize`, `const _CharT * __first`, `const _CharT * __last` )
- template<typename \_Tp >  
`constexpr _Tp * __addressof` ( `_Tp & __r` ) noexcept
- template<typename \_ForwardIterator, typename \_BinaryPredicate >  
`constexpr _ForwardIterator __adjacent_find` ( `_ForwardIterator __first`, `_ForwardIterator __last`, `_BinaryPredicate __binary_pred` )
- template<typename \_RandomAccessIterator, typename \_Distance, typename \_Tp, typename \_Compare >  
`constexpr void __adjust_heap` ( `_RandomAccessIterator __first`, `_Distance __holeIndex`, `_Distance __len`, `_Tp __value`, `_Compare __comp` )
- template<typename \_InputIterator, typename \_Distance >  
`constexpr void __advance` ( `_InputIterator & __i`, `_Distance __n`, `input_iterator_tag` )
- template<typename \_BidirectionalIterator, typename \_Distance >  
`constexpr void __advance` ( `_BidirectionalIterator & __i`, `_Distance __n`, `bidirectional_iterator_tag` )
- template<typename \_RandomAccessIterator, typename \_Distance >  
`constexpr void __advance` ( `_RandomAccessIterator & __i`, `_Distance __n`, `random_access_iterator_tag` )
- template<typename \_Alloc >  
`constexpr void __alloc_on_copy` ( `_Alloc & __one`, `const _Alloc & __two` )
- template<typename \_Alloc >  
`constexpr _Alloc __alloc_on_copy` ( `const _Alloc & __a` )
- template<typename \_Alloc >  
`constexpr void __alloc_on_move` ( `_Alloc & __one`, `_Alloc & __two` )
- template<typename \_Alloc >  
`constexpr void __alloc_on_swap` ( `_Alloc & __one`, `_Alloc & __two` )
- template<typename \_Alloc >  
`__allocated_ptr`< `_Alloc` > `__allocate_guarded` ( `_Alloc & __a` )
- template<typename \_Tp, \_Lock\_policy \_Lp = \_\_default\_lock\_policy, typename \_Alloc, typename... \_Args >  
`__shared_ptr`< `_Tp`, `_Lp` > `__allocate_shared` ( `const _Alloc & __a`, `_Args &&... __args` )

- `template<typename _Fn, typename _Tp, typename... _Args>`  
`constexpr bool __call_is_nt (__invoke_memfun_ref)`
- `template<typename _Fn, typename _Tp, typename... _Args>`  
`constexpr bool __call_is_nt (__invoke_memfun_deref)`
- `template<typename _Fn, typename _Tp >`  
`constexpr bool __call_is_nt (__invoke_memobj_ref)`
- `template<typename _Fn, typename _Tp >`  
`constexpr bool __call_is_nt (__invoke_memobj_deref)`
- `template<typename _Fn, typename... _Args>`  
`constexpr bool __call_is_nt (__invoke_other)`
- `template<typename _Facet >`  
`const _Facet & __check_facet (const _Facet * __f)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`constexpr void __chunk_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)`
- `constexpr memory_order __cmpexch_failure_order (memory_order __m) noexcept`
- `constexpr memory_order __cmpexch_failure_order2 (memory_order __m) noexcept`
- `template<typename _Tp >`  
`_Tp __complex_abs (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acos (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_acosh (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`_Tp __complex_arg (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asin (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_asinh (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atan (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_atanh (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cos (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_cosh (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_exp (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_log (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow (const complex< _Tp > & __x, const complex< _Tp > & __y)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp >`  
`std::complex< _Tp > __complex_proj (const std::complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sin (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sinh (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_sqrt (const complex< _Tp > & __z)`

- `template<typename _Tp >`  
`complex< _Tp > __complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > __complex_tanh (const complex< _Tp > &__z)`
- `int __convert_from_v (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)`
- `template<typename _Tp >`  
`void __convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void __convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI __copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI __copy_move_a (const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > __copy_move_a (_II __first, _II __last, const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _Ite, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > __copy_move_a (const ::__gnu_debug::Safe_iterator< _Ite, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _Ite, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI __copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI __copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > __copy_move_a1 (::Deque_iterator< _ITp, _IRef, _IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`__gnu_cxx::enable_if< __is_random_access_iter< _II >::value, ::Deque_iterator< _Tp, _Tp &, _Tp * > >::__type __copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT > >::__type __copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type __copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > > >)`

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type __copy_move_a2 (istreambuf_iterator<`  
`_CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI __copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI __copy_move_backward_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI __copy_move_backward_a (const ::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const <`  
`::__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`__gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > __copy_move_backward_a (_II, _II, const ::__gnu_debug::Safe_iterator<`  
`_Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > __copy_move_backward_a (const ::__gnu_debug::Safe_iterator<`  
`_IIte, _ISeq, _ICat > &, const ::__gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const <`  
`::__gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`constexpr _BI2 __copy_move_backward_a1 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI __copy_move_backward_a1 (:: Deque_iterator< _Tp, _Ref, _Ptr > __first, :: Deque_iterator< _Tp, _Ref,`  
`_Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`:: Deque_iterator< _OTp, _OTp &, _OTp * > __copy_move_backward_a1 (:: Deque_iterator< _ITp, _IRef,`  
`_IPtr > __first, :: Deque_iterator< _ITp, _IRef, _IPtr > __last, :: Deque_iterator< _OTp, _OTp &, _OTp * >`  
`__result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, :: Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type __copy_move_backward_a1 (_II __first, _II __last, :: Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`constexpr _BI2 __copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI __copy_move_backward_dit (:: Deque_iterator< _Tp, _Ref, _Ptr > __first, :: Deque_iterator< _Tp, _Ref,`  
`_Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI __copy_move_dit (:: Deque_iterator< _Tp, _Ref, _Ptr > __first, :: Deque_iterator< _Tp, _Ref, _Ptr >`  
`__last, _OI __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`constexpr _OutputIterator __copy_n (_InputIterator __first, _Size __n, _OutputIterator __result, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Size, typename _OutputIterator >`  
`constexpr _OutputIterator __copy_n (_RandomAccessIterator __first, _Size __n, _OutputIterator __result,`  
`random_access_iterator_tag)`
- `template<typename _CharT, typename _Size >`  
`__enable_if_t< __is_char< _CharT >::__value, _CharT * > __copy_n_a (istreambuf_iterator< _CharT > __it,`  
`_Size __n, _CharT * __result)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`constexpr _OutputIterator __copy_n_a (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _CharT, typename _Size >`  
`__enable_if_t< __is_char< _CharT >::__value, _CharT * > __copy_n_a (istreambuf_iterator< _CharT,`  
`char_traits< _CharT > >, _Size, _CharT *)`
- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs (basic_streambuf< _CharT, _Traits > * __sbin, basic_streambuf< _CharT, <`  
`_Traits > * __sbout)`

- `template<typename _CharT, typename _Traits >`  
`streamsize __copy_streambufs_eof (basic_streambuf< _CharT, _Traits > *, basic_streambuf< _CharT, _Traits > *, bool &)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< char > * __sbin, basic_streambuf< char > * __sbout, bool & __ineof)`
- `template<>`  
`streamsize __copy_streambufs_eof (basic_streambuf< wchar_t > * __sbin, basic_streambuf< wchar_t > * __sbout, bool & __ineof)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr iterator_traits< _InputIterator >::difference_type __count_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Signature, typename _Fn, typename _Alloc = std::allocator<int>>`  
`static shared_ptr< _future_base::Task_state_base< _Signature > > __create_task_state ( _Fn && __fn, const _Alloc & __a= _Alloc())`
- `constexpr size_t __deque_buf_size (size_t __size)`
- `template<typename _InputIterator >`  
`constexpr iterator_traits< _InputIterator >::difference_type __distance ( _InputIterator __first, _InputIterator __last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`constexpr iterator_traits< _RandomAccessIterator >::difference_type __distance ( _RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _Alloc >`  
`void __do_alloc_on_copy ( _Alloc & __one, const _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_copy ( _Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_move ( _Alloc & __one, _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_move ( _Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_swap ( _Alloc & __one, _Alloc & __two, true_type)`
- `template<typename _Alloc >`  
`void __do_alloc_on_swap ( _Alloc &, _Alloc &, false_type)`
- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn >`  
`bool __do_str_codecvt (const _InChar * __first, const _InChar * __last, _OutStr & __outstr, const _Codecvt & __cvt, _State & __state, size_t & __count, _Fn __fn)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool __equal4 ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _BinaryPredicate >`  
`constexpr bool __equal4 ( _II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool __equal_aux ( _II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2 >`  
`bool __equal_aux (const ::__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)`
- `template<typename _II1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool __equal_aux ( _II1, _II1, const ::__gnu_debug::Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool __equal_aux (const ::__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool __equal_aux1 ( _II1 __first1, _II1 __last1, _II2 __first2)`

- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type __equal_aux1 (↵`  
`::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2 >`  
`bool __equal_aux1 (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1`  
`> __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type __equal_aux1 (_II __first1,`  
`_II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II >`  
`bool __equal_dit (const ::_Deque_iterator< _Tp, _Ref, _Ptr > & __first1, const ::_Deque_iterator< _Tp, _Ref,`  
`_Ptr > & __last1, _II __first2)`
- `template<typename _ForwardIterator, typename _Tp, typename _CompareItTp, typename _CompareTplt >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > __equal_range (_ForwardIterator __first, _ForwardIterator`  
`__last, const _Tp & __val, _CompareItTp __comp_it_val, _CompareTplt __comp_val_it)`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp __exchange (_Tp & __obj, _Up && __new_val)`
- `template<typename _Flte, typename _Tp >`  
`constexpr void __fill_a (_Flte __first, _Flte __last, const _Tp & __value)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Tp >`  
`void __fill_a (const ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > &, const ::__gnu_debug::__Safe_iterator<`  
`_Lte, _Seq, _Cat > &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, void >::__type __fill_a1 (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, void >::__type __fill_a1 (_ForwardIterator`  
`__first, _ForwardIterator __last, const _Tp & __value)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type __fill_a1 (_Tp * __first, _Tp *↵`  
`__last, const _Tp & __c)`
- `template<typename _Lte, typename _Cont, typename _Tp >`  
`constexpr void __fill_a1 (::__gnu_cxx::__normal_iterator< _Lte, _Cont > __first, ::__gnu_cxx::__normal_↵`  
`iterator< _Lte, _Cont > __last, const _Tp & __value)`
- `template<typename _Tp, typename _VTp >`  
`void __fill_a1 (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > & __first, const ::_Deque_iterator< _Tp, _Tp &,`  
`_Tp * > & __last, const _VTp & __value)`
- `void __fill_bvector (_Bit_type * __v, unsigned int __first, unsigned int __last, bool __x)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Size, typename _Tp >`  
`::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > __fill_n_a (const ::__gnu_debug::__Safe_iterator< _Lte, ↵`  
`_Seq, _Cat > & __first, _Size __n, const _Tp & __value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value, std::output_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator __fill_n_a (_OutputIterator __first, _Size __n, const _Tp & __value, std::random_access_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, _OutputIterator >::__type __fill_n_a1 (↵`  
`_OutputIterator __first, _Size __n, const _Tp & __value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, _OutputIterator >::__type __fill_n_a1 (↵`  
`_OutputIterator __first, _Size __n, const _Tp & __value)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void \_\_final\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr _ForwardIterator1 \_\_find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward\_iterator\_tag, forward\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate >`  
`constexpr _BidirectionalIterator1 \_\_find\_end (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr _InputIterator \_\_find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`constexpr _RandomAccessIterator \_\_find\_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iterator, typename _Predicate >`  
`constexpr _Iterator \_\_find\_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr _InputIterator \_\_find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate, typename _Distance >`  
`constexpr _InputIterator \_\_find\_if\_not\_n (_InputIterator __first, _Distance &__len, _Predicate __pred)`
- `template<typename _EuclideanRingElement >`  
`constexpr _EuclideanRingElement \_\_gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)`
- `template<typename _IntType, typename _UniformRandomBitGenerator >`  
`pair< _IntType, _IntType > \_\_gen\_two\_uniform\_ints (_IntType __b0, _IntType __b1, _UniformRandomBitGenerator &&__g)`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & \_\_get\_helper (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & \_\_get\_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr _Head & \_\_get\_helper2 (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr const _Head & \_\_get\_helper2 (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void \_\_heap\_select (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`size_t \_\_iconv\_adapter (size_t>(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`constexpr bool \_\_includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void \_\_inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void \_\_inplace\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void \_\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`



- `template<typename _CharT, typename _ValueT >`  
`int __int_to_char ( _CharT * __bufend, _ValueT __v, const _CharT * __lit, ios\_base::fmtflags __flags, bool __dec)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`constexpr void __introselct ( _RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`constexpr void __introsort_loop ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && __invfwd (typename remove\_reference<_Tp>::type & __t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result< _Callable, _Args... >::type __invoke ( _Callable && __fn, _Args &&... __args) noexcept( \_is\_nothrow\_invocable< _Callable, _Args... >::value)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_other, _Fn && __f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_memfun_ref, _MemFun && __f, _Tp && __t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res __invoke_impl ( __invoke_memfun_deref, _MemFun && __f, _Tp && __t, _Args &&... __args)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res __invoke_impl ( __invoke_memobj_ref, _MemPtr && __f, _Tp && __t)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res __invoke_impl ( __invoke_memobj_deref, _MemPtr && __f, _Tp && __t)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr __can_invoke_as_nonvoid< _Res, _Callable, _Args... > __invoke_r ( _Callable && __fn, _Args &&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr __can_invoke_as_void< _Res, _Callable, _Args... > __invoke_r ( _Callable && __fn, _Args &&... __args)`
- `template<typename _Tp, size_t = sizeof(_Tp)>`  
`constexpr true\_type __is_complete_or_unbounded ( __type_identity< _Tp >)`
- `template<typename _TypelIdentity, typename _NestedType = typename _TypelIdentity::type>`  
`constexpr __or< is\_reference< _NestedType >, is\_function< _NestedType >, is\_void< _NestedType >, is\_array\_unknown\_bounds< _NestedType > >::type __is_complete_or_unbounded ( _TypelIdentity)`
- `template<typename _Ch, typename _Up >`  
`basic\_istream< _Ch, _Up > & __is_convertible_to_basic_istream_test (basic\_istream< _Ch, _Up > *)`
- `template<typename _Ch, typename _Up >`  
`basic\_ostream< _Ch, _Up > & __is_convertible_to_basic_ostream_test (basic\_ostream< _Ch, _Up > *)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance >`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator >`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr bool __is_heap ( _RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare >`  
`constexpr _Distance __is_heap_until ( _RandomAccessIterator __first, _Distance __n, _Compare & __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool __is_permutation ( _ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`



- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool __is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator __is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iter >`  
`constexpr iterator\_traits<_Iter>::iterator_category __iterator_category (const _Iter &)`
- `template<typename _I1, typename _I2 >`  
`constexpr bool __lexicographical_compare_aux (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare >`  
`constexpr bool __lexicographical_compare_impl (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, __Compare __comp)`
- `constexpr int __lg (int __n)`
- `constexpr unsigned __lg (unsigned __n)`
- `constexpr long __lg (long __n)`
- `constexpr unsigned long __lg (unsigned long __n)`
- `constexpr long long __lg (long long __n)`
- `constexpr unsigned long long __lg (unsigned long long __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator __lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond<typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = typename conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>::type>`  
`constexpr _ReturnType __make_move_if_noexcept_iterator (_Tp * __i)`
- `template<typename _Iterator >`  
`constexpr reverse\_iterator<_Iterator> __make_reverse_iterator (_Iterator __i)`
- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>`  
`__shared_ptr<_Tp, _Lp> __make_shared (_Args &&... __args)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator __max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Up >`  
`constexpr int __memcmp (const _Tp * __first1, const _Up * __first2, size_t __num)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator __merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`  
`void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`  
`void __merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`  
`void __merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`

- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`  
`void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator`  
`__last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator __min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > __minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > __mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > __mismatch (_InputIterator1 __first1, _InputIterator1 __last1,`  
`_InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator >`  
`constexpr _Iterator __miter_base (_Iterator __it)`
- `template<typename _Iterator >`  
`constexpr auto __miter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__miter_base(__it.base())))`
- `template<typename _Iterator >`  
`auto __miter_base (move_iterator< _Iterator > __it) -> decltype(__miter_base(__it.base()))`
- `template<typename _Iterator, typename _Compare >`  
`constexpr void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2,`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2,`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`  
`void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2,`  
`_BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool __next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Iterator >`  
`constexpr _Iterator __niter_base (_Iterator __it) noexcept(/*conditional */)`
- `template<typename _Iterator >`  
`constexpr auto __niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__niter_base(__it.base())))`
- `template<typename _Iterator, typename _Container >`  
`constexpr _Iterator __niter_base (__gnu_cxx::__normal_iterator< _Iterator, _Container > __it) noexcept(/*conditional */)`
- `template<typename _Iterator >`  
`auto __niter_base (move_iterator< _Iterator > __it) -> decltype(make_move_iterator(__niter_base(__it.base())))`
- `template<typename _From, typename _To >`  
`constexpr _From __niter_wrap (_From __from, _To __res)`
- `template<typename _Iterator >`  
`constexpr _Iterator __niter_wrap (const _Iterator &, _Iterator __res)`

- `template<typename _CharT, typename _Traits >`  
`void __ostream_fill (basic\_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`basic\_ostream< _CharT, _Traits > & __ostream_insert (basic\_ostream< _CharT, _Traits > &__out, const _↵  
CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`void __ostream_write (basic\_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↵  
AccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator __partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↵  
AccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred,  
forward\_iterator\_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`constexpr _BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate  
__pred, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Random↵  
AccessIterator __result, _Compare &__comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool __prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __↵  
comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`constexpr void __push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex,  
_Tp __value, _Compare &__comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`constexpr _OutputIterator __remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator __remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`constexpr _OutputIterator __replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↵  
result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`constexpr void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional\_iterator\_tag)`
- `template<typename _RandomAccessIterator >`  
`constexpr void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random\_access\_iterator\_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _↵  
BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __↵  
__buffer_size)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator >`  
`_RandomAccessIterator __sample (_InputIterator __first, _InputIterator __last, input\_iterator\_tag, _Random↵  
AccessIterator __out, random\_access\_iterator\_tag, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↵  
Generator >`  
`_OutputIterator __sample (_ForwardIterator __first, _ForwardIterator __last, forward\_iterator\_tag, _OutputIterator  
__out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr _ForwardIterator1 __search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2  
__first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`

- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`constexpr _ForwardIterator __search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate >`  
`constexpr _ForwardIterator __search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate >`  
`constexpr _RandomAccessIter __search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator __set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator __set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator __set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator __set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `constexpr int __size_to_integer (int __n)`
- `constexpr unsigned __size_to_integer (unsigned __n)`
- `constexpr long __size_to_integer (long __n)`
- `constexpr unsigned long __size_to_integer (unsigned long __n)`
- `constexpr long long __size_to_integer (long long __n)`
- `constexpr unsigned long long __size_to_integer (unsigned long long __n)`
- `constexpr long long __size_to_integer (float __n)`
- `constexpr long long __size_to_integer (double __n)`
- `constexpr long long __size_to_integer (long double __n)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare & __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator __stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance >`  
`_ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void __stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare >`  
`void __stable_sort_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt, _State & __state, size_t & __count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_in (const char * __first, const char * __last, basic\_string< _CharT, _Traits, _Alloc > & __outstr, const codecvt< _CharT, char, _State > & __cvt)`

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_in_all (const char *__first, const char *__last, basic\_string< _CharT, _Traits, _Alloc > &↵`  
`__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_out (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc >`  
`&__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_out (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc >`  
`&__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State >`  
`bool __str_codecvt_out_all (const _CharT *__first, const _CharT *__last, basic\_string< char, _Traits, _Alloc >`  
`&__outstr, const codecvt< _CharT, char, _State > &__cvt)`
- `void __throw_bad_alloc (void)`
- `void __throw_bad_cast (void)`
- `void __throw_bad_exception (void)`
- `void __throw_bad_function_call ()`
- `void __throw_bad_typeid (void)`
- `void __throw_bad_weak_ptr ()`
- `void __throw_domain_error (const char *)`
- `void __throw_future_error (int)`
- `void __throw_invalid_argument (const char *)`
- `void __throw_ios_failure (const char *)`
- `void __throw_ios_failure (const char *, int)`
- `void __throw_length_error (const char *)`
- `void __throw_logic_error (const char *)`
- `void __throw_out_of_range (const char *)`
- `void __throw_out_of_range_fmt (const char *,...)`
- `void __throw_overflow_error (const char *)`
- `void __throw_range_error (const char *)`
- `void __throw_regex_error (regex\_constants::error\_type __ecode)`
- `void __throw_regex_error (regex\_constants::error\_type __ecode, const char *__what)`
- `void __throw_runtime_error (const char *)`
- `void __throw_system_error (int)`
- `void __throw_underflow_error (const char *)`
- `template<typename _Tp >`  
`constexpr _Tp * __to_address (_Tp *__ptr) noexcept`
- `template<typename _Ptr >`  
`constexpr std::pointer\_traits< _Ptr >::element_type * __to_address (const _Ptr &__ptr)`
- `template<typename _Tp >`  
`__detail::__integer_to_chars_result_type< _Tp > __to_chars_i (char *__first, char *__last, _Tp __value, int`  
`__base=10)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __unguarded_insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, ↵`  
`_Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void __unguarded_linear_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator __unguarded_partition (_RandomAccessIterator __first, _RandomAccess↵`  
`Iterator __last, _RandomAccessIterator __pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator __unguarded_partition_pivot (_RandomAccessIterator __first, _Random↵`  
`AccessIterator __last, _Compare __comp)`

- `template<typename _Pointer, typename _ForwardIterator >`  
`void __uninitialized_construct_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator __unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator __unique_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __↵`  
`result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator __unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator __unique_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result,`  
`_BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator __upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,`  
`_Compare __comp)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr __uses_alloc_t< _Tp, _Alloc, _Args... > __use_alloc (const _Alloc &__a)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __use_alloc (const _Alloc &&)=delete`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct (const _Alloc &__a, _Tp *__ptr, _Args &&... __args)`
- `template<typename _Tp, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc0 __a, _Tp *__ptr, _Args &&... __args)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc1< _Alloc > __a, _Tp *__ptr, _Args &&... __args)`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`void __uses_allocator_construct_impl (__uses_alloc2< _Alloc > __a, _Tp *__ptr, _Args &&... __args)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __k)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t >`  
`__i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array<`  
`size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`

- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void __valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void __valarray_copy (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void __valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void __valarray_copy_construct (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void __valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`

- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill (_Array<_Tp> __a, _Array<size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp __t)`
- `template<typename _Tp >`  
`_Tp * __valarray_get_storage (size_t)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type __valarray_min (const _Ta &__a)`
- `void __valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp __valarray_sum (const _Tp *__f, const _Tp *__l)`
- `bool __verify_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (volatile tuple<_Tp...> &__tuple) -> __tuple_element_t<_Ind, tuple<_Tp...>> volatile &`
- `template<std::size_t _Ind, typename... _Tp>`  
`auto __volget (const volatile tuple<_Tp...> &__tuple) -> __tuple_element_t<_Ind, tuple<_Tp...>> const volatile &`
- `template<typename _CharT >`  
`ostreambuf_iterator<_CharT> __write (ostreambuf_iterator<_CharT> __s, const _CharT *__ws, int __len)`
- `template<typename _CharT, typename _Outlter >`  
`_Outlter __write (_Outlter __s, const _CharT *__ws, int __len)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<bool> __m)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, const Expr<_Dom, _Tp> &__e, size_t __n)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void __Array_augmented__bitwise_and (_Array<_Tp> __a, size_t __s, const Expr<_Dom, _Tp> &__e, size_t __n)`



- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void _Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented_shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void _Array_augmented_shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`

- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void _Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, typename... _Args>`  
`void _Construct (_Tp *__p, _Args &&... __args)`
- `template<typename _T1 >`  
`void _Construct_novalue (_T1 *__p)`
- `template<typename _ForwardIterator >`  
`constexpr void _Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp >`  
`constexpr void _Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void _Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`
- `template<typename _ForwardIterator, typename _Size >`  
`constexpr _ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)`
- `size_t _Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `template<typename _Tp >`  
`std::__is_nullptr_t is_null_pointer _GLIBCXX_DEPRECATED_SUGGEST ("std::is_null_pointer")`
- `size_t _Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `unsigned int _Rb_tree_black_count (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__↵  
root) throw ()`
- `_Rb_tree_node_base * _Rb_tree_decrement (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_decrement (const _Rb_tree_node_base *__x) throw ()`
- `_Rb_tree_node_base * _Rb_tree_increment (_Rb_tree_node_base *__x) throw ()`
- `const _Rb_tree_node_base * _Rb_tree_increment (const _Rb_tree_node_base *__x) throw ()`
- `void _Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node↵  
_base *__p, _Rb_tree_node_base &__header) throw ()`
- `_Rb_tree_node_base * _Rb_tree_rebalance_for_erase (_Rb_tree_node_base *const __z, _Rb_tree_node↵  
_base &__header) throw ()`
- `void abort (void) throw ()`
- `long abs (long __i)`
- `long long abs (long long __x)`

- constexpr double **abs** (double \_\_x)
- template<typename \_Tp >  
\_Tp **abs** (const complex<\_Tp> &)
- constexpr float **abs** (float \_\_x)
- constexpr long double **abs** (long double \_\_x)
- template<class \_Dom >  
\_Expr<\_UnClos< struct std::\_Abs, \_Expr, \_Dom >, typename \_Dom::value\_type > **abs** (const \_Expr<\_Dom, typename \_Dom::value\_type> &\_\_e)
- template<typename \_Tp >  
\_Expr<\_UnClos< struct std::\_Abs, \_ValArray, \_Tp >, \_Tp > **abs** (const valarray<\_Tp> &\_\_v)
- template<typename \_InputIterator, typename \_Tp >  
constexpr \_Tp **accumulate** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init)
- template<typename \_InputIterator, typename \_Tp, typename \_BinaryOperation >  
constexpr \_Tp **accumulate** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op)
- constexpr float **acos** (float \_\_x)
- constexpr long double **acos** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if<\_\_is\_integer<\_Tp>::\_\_value, double>::\_\_type **acos** (\_Tp \_\_x)
- template<class \_Dom >  
\_Expr<\_UnClos< struct std::\_Acos, \_Expr, \_Dom >, typename \_Dom::value\_type > **acos** (const \_Expr<\_Dom, typename \_Dom::value\_type> &\_\_e)
- template<typename \_Tp >  
\_Expr<\_UnClos< struct std::\_Acos, \_ValArray, \_Tp >, \_Tp > **acos** (const valarray<\_Tp> &\_\_v)
- template<typename \_Tp >  
**std::complex**<\_Tp> **acos** (const std::complex<\_Tp> &\_\_z)
- template<typename \_Tp >  
**std::complex**<\_Tp> **acosh** (const std::complex<\_Tp> &\_\_z)
- template<typename \_Tp >  
constexpr \_Tp \* **addressof** (\_Tp &\_\_r) noexcept
- template<typename \_Tp >  
const \_Tp \* **addressof** (const \_Tp &&) = delete
- template<typename \_InputIterator, typename \_OutputIterator >  
constexpr \_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_BinaryOperation >  
constexpr \_OutputIterator **adjacent\_difference** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, \_BinaryOperation \_\_binary\_op)
- template<typename \_Filter >  
constexpr \_Filter **adjacent\_find** (\_Filter, \_Filter)
- template<typename \_Filter, typename \_BinaryPredicate >  
constexpr \_Filter **adjacent\_find** (\_Filter, \_Filter, \_BinaryPredicate)
- template<typename \_ForwardIterator >  
constexpr \_ForwardIterator **adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_ForwardIterator, typename \_BinaryPredicate >  
constexpr \_ForwardIterator **adjacent\_find** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_BinaryPredicate \_\_binary\_pred)
- template<typename \_InputIterator, typename \_Distance >  
constexpr void **advance** (\_InputIterator &\_\_i, \_Distance \_\_n)
- template<typename \_CharT, typename \_Distance >  
\_\_gnu\_cxx::\_\_enable\_if<\_\_is\_char<\_CharT>::\_\_value, void>::\_\_type **advance** (istreambuf\_iterator<\_CharT> &\_\_i, \_Distance \_\_n)
- void \* **align** (size\_t \_\_align, size\_t \_\_size, void \*&\_\_ptr, size\_t &\_\_space) noexcept

- `template<typename _Iter, typename _Predicate >`  
`constexpr bool all_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr bool any_of (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`_Tp arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type arg (_Tp __x)`
- `constexpr float asin (float __x)`
- `constexpr long double asin (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type asin (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > asin (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > asin (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`std::complex< _Tp > asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (_Fn &&__fn, _Args &&... __args)`
- `constexpr float atan (float __x)`
- `constexpr long double atan (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type atan (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > atan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > atan (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`std::complex< _Tp > atan (const std::complex< _Tp > &__z)`
- `constexpr float atan2 (float __y, float __x)`
- `constexpr long double atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type atan2 (_Tp __y, _Up __x)`



- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > atan2 (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > atan2 (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`std::complex< _Tp > atanh (const std::complex< _Tp > &__z)`
- `int atexit (void(*) (void)) throw ()`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`

- `template<typename _ITp >`  
`bool atomic_compare_exchange_weak_explicit (volatile atomic<_ITp > *__a, __atomic_val_t<_ITp > *__i1, __atomic_val_t<_ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (atomic<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange (volatile atomic<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (atomic<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_exchange_explicit (volatile atomic<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add (volatile atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_add_explicit (volatile atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and (__atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and (volatile __atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (__atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_and_explicit (volatile __atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (__atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or (volatile __atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (__atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_or_explicit (volatile __atomic_base<_ITp > *__a, __atomic_val_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub (volatile atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_sub_explicit (volatile atomic<_ITp > *__a, __atomic_diff_t<_ITp > __i, memory_order __m) noexcept`

- `template<typename _ITp >`  
`_ITp atomic_fetch_xor ( __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit ( __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_flag_clear (atomic_flag *__a) noexcept`
- `void atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`bool atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`bool atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `void atomic_signal_fence (memory_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_store (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void atomic_store (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void atomic_store_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void atomic_store_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void atomic_thread_fence (memory_order __m) noexcept`
- `template<typename _Container >`  
`constexpr back_insert_iterator< _Container > back_inserter (_Container &__x)`
- `template<typename _Container >`  
`constexpr auto begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Container >`  
`constexpr auto begin (const _Container &__cont) -> decltype(__cont.begin())`

- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * begin (_Tp(&__arr)[_Nm]) noexcept`
- `template<class _Tp >`  
`_Tp * begin (valarray< _Tp > &__va)`
- `template<class _Tp >`  
`const _Tp * begin (const valarray< _Tp > &__va)`
- `template<typename _Tpa, typename _Tpb >`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type beta (_Tpa __a, _Tpb __b)`
- `float betaf (float __a, float __b)`
- `long double betal (long double __a, long double __b)`
- `template<typename _Filter, typename _Tp >`  
`constexpr bool binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr bool binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`
- `template<typename _Func, typename... _BoundArgs>`  
`constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>`  
`constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args) ↵`
- `template<typename _Operation, typename _Tp >`  
`binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation, typename _Tp >`  
`binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)`
- `ios_base & boolalpha (ios_base & __base)`
- `template<typename _Callable, typename... _Args>`  
`void call_once (once_flag & __once, _Callable &&__f, _Args &&... __args)`
- `template<typename _Container >`  
`constexpr auto cbegin (const _Container & __cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(↵  
__cont))`
- `constexpr float ceil (float __x)`
- `constexpr long double ceil (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ceil (_Tp __x)`
- `template<typename _Container >`  
`constexpr auto cend (const _Container & __cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(↵  
__cont))`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)`
- `float comp_ellint_1f (float __k)`
- `long double comp_ellint_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)`
- `float comp_ellint_2f (float __k)`
- `long double comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)`

- float `comp_ellint_3f` (float \_\_k, float \_\_nu)
- long double `comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- template<typename \_Tp >  
constexpr `complex`< \_Tp > `conj` (const `complex`< \_Tp > &)
- template<typename \_Tp >  
constexpr `std::complex`< typename \_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type > `conj` (\_Tp \_\_x)
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy\_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > `const_pointer_cast` (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_r) noexcept
- template<typename \_Iter, typename \_OIter >  
constexpr \_OIter `copy` (\_Iter, \_Iter, \_OIter)
- template<typename \_CharT >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, ostreambuf\_iterator< \_CharT >::\_\_type > `copy`  
(istreambuf\_iterator< \_CharT > \_\_first, istreambuf\_iterator< \_CharT > \_\_last, ostreambuf\_iterator< \_CharT >  
> \_\_result)
- template<typename \_II, typename \_OI >  
constexpr \_OI `copy` (\_II \_\_first, \_II \_\_last, \_OI \_\_result)
- template<typename \_BIter1, typename \_BIter2 >  
constexpr \_BIter2 `copy_backward` (\_BIter1, \_BIter1, \_BIter2)
- template<typename \_BI1, typename \_BI2 >  
constexpr \_BI2 `copy_backward` (\_BI1 \_\_first, \_BI1 \_\_last, \_BI2 \_\_result)
- template<typename \_Iter, typename \_OIter, typename \_Predicate >  
constexpr \_OIter `copy_if` (\_Iter, \_Iter, \_OIter, \_Predicate)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_Predicate >  
constexpr \_OutputIterator `copy_if` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, \_Predicate  
\_\_pred)
- template<typename \_Iter, typename \_Size, typename \_OIter >  
constexpr \_OIter `copy_n` (\_Iter, \_Size, \_OIter)
- template<typename \_InputIterator, typename \_Size, typename \_OutputIterator >  
constexpr \_OutputIterator `copy_n` (\_InputIterator \_\_first, \_Size \_\_n, \_OutputIterator \_\_result)
- template<typename \_Tp >  
`complex`< \_Tp > `cos` (const `complex`< \_Tp > &)
- constexpr float `cos` (float \_\_x)
- constexpr long double `cos` (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type `cos` (\_Tp \_\_x)
- template<typename \_Tp >  
\_Expr< \_UnClos< struct std::\_Cos, \_ValArray, \_Tp >, \_Tp > `cos` (const `valarray`< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< struct std::\_Cos, \_Expr, \_Dom >, typename \_Dom::value\_type > `cos` (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e)
- template<typename \_Tp >  
`complex`< \_Tp > `cosh` (const `complex`< \_Tp > &)
- constexpr float `cosh` (float \_\_x)
- constexpr long double `cosh` (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type `cosh` (\_Tp \_\_x)
- template<typename \_Tp >  
\_Expr< \_UnClos< struct std::\_Cosh, \_ValArray, \_Tp >, \_Tp > `cosh` (const `valarray`< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< struct std::\_Cosh, \_Expr, \_Dom >, typename \_Dom::value\_type > `cosh` (const \_Expr< \_Dom,  
typename \_Dom::value\_type > &\_\_e)
- template<typename \_Iter, typename \_Tp >  
constexpr `iterator_traits`< \_Iter >::difference\_type `count` (\_Iter, \_Iter, const \_Tp &)

- `template<typename _InputIterator, typename _Tp >`  
`constexpr iterator_traits< _InputIterator >::difference_type count ( _InputIterator __first, _InputIterator __last,`  
`const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr iterator_traits< _Iter >::difference_type count_if ( _Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr iterator_traits< _InputIterator >::difference_type count_if ( _InputIterator __first, _InputIterator __last,`  
`_Predicate __pred)`
- `template<typename _Container >`  
`constexpr auto crbegin (const _Container &__cont) -> decltype(std::rbegin(__cont))`
- `template<typename _Container >`  
`constexpr auto crend (const _Container &__cont) -> decltype(std::rend(__cont))`
- `exception_ptr current_exception () noexcept`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i ( _Tpnu __nu, _Tp __x)`
- `float cyl_bessel_if (float __nu, float __x)`
- `long double cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j ( _Tpnu __nu, _Tp __x)`
- `float cyl_bessel_jf (float __nu, float __x)`
- `long double cyl_bessel_jl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float cyl_bessel_kf (float __nu, float __x)`
- `long double cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float cyl_neumannf (float __nu, float __x)`
- `long double cyl_neumannl (long double __nu, long double __x)`
- `ios_base & dec (ios_base &__base)`
- `void declare_no_pointers (char *, size_t)`
- `void declare_reachable (void *)`
- `template<typename _Tp >`  
`auto declval () noexcept -> decltype(__declval< _Tp >())`
- `ios_base & defaultfloat (ios_base &__base)`
- `template<typename _InputIterator >`  
`constexpr iterator_traits< _InputIterator >::difference_type distance ( _InputIterator __first, _InputIterator __last)`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>`  
`__shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 ( _Tp __k, _Tpp __phi)`
- `float ellint_1f (float __k, float __phi)`
- `long double ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 ( _Tp __k, _Tpp __phi)`
- `float ellint_2f (float __k, float __phi)`
- `long double ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float ellint_3f (float __k, float __nu, float __phi)`
- `long double ellint_3l (long double __k, long double __nu, long double __phi)`

- `template<typename _Container >`  
`constexpr auto end (_Container &__cont) -> decltype(__cont.end())`
- `template<typename _Container >`  
`constexpr auto end (const _Container &__cont) -> decltype(__cont.end())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * end (_Tp(&__arr)[_Nm]) noexcept`
- `template<class _Tp >`  
`_Tp * end (valarray< _Tp > &__va)`
- `template<class _Tp >`  
`const _Tp * end (const valarray< _Tp > &__va)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > &__os)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _I1, typename _I2 >`  
`constexpr bool equal (_I1 __first1, _I1 __last1, _I2 __first2)`
- `template<typename _I1, typename _I2 >`  
`constexpr bool equal (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr bool equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`constexpr pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr pair< _Filter, _Filter > equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp exchange (_Tp &__obj, _Up &&__new_val)`
- `void exit (int) throw ()`
- `template<typename _Tp >`  
`complex< _Tp > exp (const complex< _Tp > &)`
- `constexpr float exp (float __x)`
- `constexpr long double exp (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type exp (_Tp __x)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > exp (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)`
- `float expintf (float __x)`

- long double [expintl](#) (long double \_\_x)
- constexpr float **fabs** (float \_\_x)
- constexpr long double **fabs** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **fabs** (\_Tp \_\_x)
- template<typename \_Tp >  
\_Tp **fabs** (const std::complex< \_Tp > &\_\_z)
- template<typename \_Filter, typename \_Tp >  
constexpr void **fill** (\_Filter, \_Filter, const \_Tp &)
- void **fill** (\_Bit\_iterator \_\_first, \_Bit\_iterator \_\_last, const bool &\_\_x)
- template<typename \_ForwardIterator, typename \_Tp >  
constexpr void **fill** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, const \_Tp &\_\_value)
- template<typename \_OIter, typename \_Size, typename \_Tp >  
constexpr \_OIter **fill\_n** (\_OIter, \_Size, const \_Tp &)
- template<typename \_OI, typename \_Size, typename \_Tp >  
constexpr \_OI **fill\_n** (\_OI \_\_first, \_Size \_\_n, const \_Tp &\_\_value)
- template<typename \_CharT >  
\_\_gnu\_cxx::\_\_enable\_if< \_\_is\_char< \_CharT >::\_\_value, istreambuf\_iterator< \_CharT > >::\_\_type **find**  
(istreambuf\_iterator< \_CharT > \_\_first, istreambuf\_iterator< \_CharT > \_\_last, const \_CharT &\_\_val)
- template<typename \_Iter, typename \_Tp >  
constexpr \_Iter **find** (\_Iter, \_Iter, const \_Tp &)
- template<typename \_InputIterator, typename \_Tp >  
constexpr \_InputIterator **find** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Tp &\_\_val)
- template<typename \_Filter1, typename \_Filter2 >  
constexpr \_Filter1 **find\_end** (\_Filter1, \_Filter1, \_Filter2, \_Filter2)
- template<typename \_Filter1, typename \_Filter2, typename \_BinaryPredicate >  
constexpr \_Filter1 **find\_end** (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_BinaryPredicate)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2 >  
constexpr \_ForwardIterator1 **find\_end** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2  
\_\_first2, \_ForwardIterator2 \_\_last2)
- template<typename \_ForwardIterator1, typename \_ForwardIterator2, typename \_BinaryPredicate >  
constexpr \_ForwardIterator1 **find\_end** (\_ForwardIterator1 \_\_first1, \_ForwardIterator1 \_\_last1, \_ForwardIterator2  
\_\_first2, \_ForwardIterator2 \_\_last2, \_BinaryPredicate \_\_comp)
- template<typename \_Filter1, typename \_Filter2 >  
constexpr \_Filter1 **find\_first\_of** (\_Filter1, \_Filter1, \_Filter2, \_Filter2)
- template<typename \_Filter1, typename \_Filter2, typename \_BinaryPredicate >  
constexpr \_Filter1 **find\_first\_of** (\_Filter1, \_Filter1, \_Filter2, \_Filter2, \_BinaryPredicate)
- template<typename \_InputIterator, typename \_ForwardIterator >  
constexpr \_InputIterator **find\_first\_of** (\_InputIterator \_\_first1, \_InputIterator \_\_last1, \_ForwardIterator \_\_first2, \_↵  
ForwardIterator \_\_last2)
- template<typename \_InputIterator, typename \_ForwardIterator, typename \_BinaryPredicate >  
constexpr \_InputIterator **find\_first\_of** (\_InputIterator \_\_first1, \_InputIterator \_\_last1, \_ForwardIterator \_\_first2, \_↵  
ForwardIterator \_\_last2, \_BinaryPredicate \_\_comp)
- template<typename \_Iter, typename \_Predicate >  
constexpr \_Iter **find\_if** (\_Iter, \_Iter, \_Predicate)
- template<typename \_InputIterator, typename \_Predicate >  
constexpr \_InputIterator **find\_if** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_Iter, typename \_Predicate >  
constexpr \_Iter **find\_if\_not** (\_Iter, \_Iter, \_Predicate)
- template<typename \_InputIterator, typename \_Predicate >  
constexpr \_InputIterator **find\_if\_not** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Predicate \_\_pred)
- **ios\_base** & **fixed** (**ios\_base** &\_\_base)



- constexpr float **floor** (float \_\_x)
- constexpr long double **floor** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **floor** (\_Tp \_\_x)
- template<typename \_CharT, typename \_Traits >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **flush** ([basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os)
- constexpr float **fmod** (float \_\_x, float \_\_y)
- constexpr long double **fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **fmod** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Iter, typename \_Funct >  
constexpr \_Funct **for\_each** (\_Iter, \_Iter, \_Funct)
- template<typename \_InputIterator, typename \_Function >  
constexpr \_Function **for\_each** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_Function \_\_f)
- template<typename \_Tp >  
constexpr \_Tp && **forward** (typename [std::remove\\_reference](#)< \_Tp >::type &\_\_t) noexcept
- template<typename \_Tp >  
constexpr \_Tp && **forward** (typename [std::remove\\_reference](#)< \_Tp >::type &&\_\_t) noexcept
- template<typename... \_Elements>  
constexpr [tuple](#)< \_Elements &&... > **forward\_as\_tuple** (\_Elements &&... \_\_args) noexcept
- float **frexp** (float \_\_x, int \* \_\_exp)
- long double **frexp** (long double \_\_x, int \* \_\_exp)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **frexp** (\_Tp \_\_x, int \* \_\_exp)
- template<typename \_Tp >  
\_\_detail::\_\_integer\_from\_chars\_result\_type< \_Tp > **from\_chars** (const char \* \_\_first, const char \* \_\_last, \_Tp & \_\_value, int \_\_base=10)
- template<typename \_Container >  
constexpr [front\\_insert\\_iterator](#)< \_Container > **front\_inserter** (\_Container &\_\_x)
- const error\_category & **future\_category** () noexcept
- template<typename \_Filter, typename \_Generator >  
constexpr void **generate** (\_Filter, \_Filter, \_Generator)
- template<typename \_ForwardIterator, typename \_Generator >  
constexpr void **generate** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Generator \_\_gen)
- template<typename \_RealType, size\_t \_\_bits, typename \_UniformRandomNumberGenerator >  
\_RealType **generate\_canonical** (\_UniformRandomNumberGenerator &\_\_g)
- template<typename \_OIter, typename \_Size, typename \_Generator >  
constexpr \_OIter **generate\_n** (\_OIter, \_Size, \_Generator)
- template<typename \_OutputIterator, typename \_Size, typename \_Generator >  
constexpr \_OutputIterator **generate\_n** (\_OutputIterator \_\_first, \_Size \_\_n, \_Generator \_\_gen)
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
constexpr [tuple\\_element](#)< \_Int, [std::pair](#)< \_Tp1, \_Tp2 > >::type & **get** ([std::pair](#)< \_Tp1, \_Tp2 > &\_\_in) noexcept
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
constexpr [tuple\\_element](#)< \_Int, [std::pair](#)< \_Tp1, \_Tp2 > >::type && **get** ([std::pair](#)< \_Tp1, \_Tp2 > &&\_\_in) noexcept
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
constexpr const [tuple\\_element](#)< \_Int, [std::pair](#)< \_Tp1, \_Tp2 > >::type & **get** (const [std::pair](#)< \_Tp1, \_Tp2 > &\_\_in) noexcept
- template<std::size\_t \_Int, class \_Tp1, class \_Tp2 >  
constexpr const [tuple\\_element](#)< \_Int, [std::pair](#)< \_Tp1, \_Tp2 > >::type && **get** (const [std::pair](#)< \_Tp1, \_Tp2 > &&\_\_in) noexcept

- `template<typename _Tp, typename _Up >`  
`constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp && get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t _i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t _i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t _i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp & get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp && get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp & get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp && get (const tuple< _Types... > &&__t) noexcept`
- `Catalogs & get_catalogs ()`
- `template<typename _Del, typename _Tp, _Lock_policy _Lp>`  
`_Del * get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`
- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)`
- `new_handler get_new_handler () noexcept`
- `pointer_safety get_pointer_safety () noexcept`
- `template<typename _Tp >`  
`pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept`
- `terminate_handler get_terminate () noexcept`

- `template<typename _CharT >`  
`_Get_time< _CharT > get\_time (std::tm *__tmb, const _CharT *__fmt)`
- `unexpected\_handler get\_unexpected () noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &&__is, basic\_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic\_istream< _CharT, _Traits > & getline (basic\_istream< _CharT, _Traits > &&__is, basic\_string< _CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic\_istream< char > & getline (basic\_istream< char > &__in, basic\_string< char > &__str, char __delim)`
- `template<>`  
`basic\_istream< wchar_t > & getline (basic\_istream< wchar_t > &__in, basic\_string< wchar_t > &__str, wchar_t __delim)`
- `template<typename _Facet >`  
`bool has\_facet (const locale &__loc) throw ()`
- `template<typename _Tp >`  
`\_\_gnu\_cxx::\_\_promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)`
- `float hermitef (unsigned int __n, float __x)`
- `long double hermitel (unsigned int __n, long double __x)`
- `ios\_base & hex (ios\_base &__base)`
- `ios\_base & hexfloat (ios\_base &__base)`
- `template<typename _Tp >`  
`constexpr _Tp imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr \_\_gnu\_cxx::\_\_promote< _Tp >::__type imag (_Tp)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`constexpr bool includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2 >`  
`constexpr _Tp inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`

- `template<typename _Blter >`  
`void inplace_merge (_Blter, _Blter, _Blter)`
- `template<typename _Blter, typename _Compare >`  
`void inplace_merge (_Blter, _Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, ↵  
_Compare __comp)`
- `template<typename _Container >`  
`insert_iterator< _Container > inserter (_Container &__x, typename _Container::iterator __i)`
- `ios_base & internal (ios_base &__base)`
- `const error_category & iostream_category () noexcept`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _RAIter >`  
`constexpr bool is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr bool is_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`constexpr _RAIter is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr _RAIter is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↵  
last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↵  
last, _Compare __comp)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr bool is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr bool is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr bool is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2,  
_ForwardIterator2 __last2, _BinaryPredicate __pred)`

- `template<typename _Filter >`  
`constexpr bool is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr bool is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Filter >`  
`constexpr _Filter is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _CharT >`  
`bool isalnum (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isalpha (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isblank (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool iscntrl (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isgraph (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool islower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isprint (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr void iter_swap (_Filter1, _Filter2)`
- `template<typename _Tp >`  
`_Tp kill_dependency (_Tp __y) noexcept`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)`
- `float laguerref (unsigned int __n, float __x)`
- `long double laguerrel (unsigned int __n, long double __x)`
- `constexpr float ldexp (float __x, int __exp)`
- `constexpr long double ldexp (long double __x, int __exp)`

- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type ldexp (_Tp __x, int __exp)`
- `ios_base & left (ios_base & __base)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __l, _Tp __x)`
- `float legendref (unsigned int __l, float __x)`
- `long double legendrel (unsigned int __l, long double __x)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`constexpr bool lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _I1, typename _I2 >`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2)`
- `template<typename _I1, typename _I2, typename _Compare >`  
`constexpr bool lexicographical_compare (_I1 __first1, _I1 __last1, _I2 __first2, _I2 __last2, _Compare __comp)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void link (_L1 & __l1, _L2 & __l2, _L3 &... __l3)`
- `template<typename _Tp >`  
`complex< _Tp > log (const complex< _Tp > &)`
- `constexpr float log (float __x)`
- `constexpr long double log (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::Log, _Expr, _Dom >, typename _Dom::value_type > log (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::Log, _ValArray, _Tp >, _Tp > log (const valarray< _Tp > & __v)`
- `template<typename _Tp >`  
`complex< _Tp > log10 (const complex< _Tp > &)`
- `constexpr float log10 (float __x)`
- `constexpr long double log10 (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type log10 (_Tp __x)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::Log10, _Expr, _Dom >, typename _Dom::value_type > log10 (const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::Log10, _ValArray, _Tp >, _Tp > log10 (const valarray< _Tp > & __v)`
- `template<typename _Filter, typename _Tp >`  
`constexpr _Filter lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr _Filter lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, ←  
_Compare __comp)`
- `error_code make_error_code (future_errc __errc) noexcept`
- `error_code make_error_code (io_errc __e) noexcept`
- `error_condition make_error_condition (future_errc __errc) noexcept`
- `error_condition make_error_condition (io_errc __e) noexcept`

- `template<typename _Ex >`  
`exception_ptr make_exception_ptr (_Ex __ex) noexcept`
- `template<typename _RandomAccessIterator >`  
`constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RAIter >`  
`constexpr void make_heap (_RAIter, _RAIter)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Iterator >`  
`constexpr move_iterator< _Iterator > make_move_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`constexpr reverse_iterator< _Iterator > make_reverse_iterator (_Iterator __i)`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::type... > make_tuple (_Elements &&... __args)`
- `template<typename _Tp >`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr _Tp max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr _Filter max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp, typename _Class >`  
`constexpr _Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::* __pm) noexcept`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp >`  
`const_mem_fun_t< _Ret, _Tp > mem_fun (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > mem_fun (_Ret(_Tp::* __f)(_Arg) const)`
- `template<typename _Ret, typename _Tp >`  
`mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)())`
- `template<typename _Ret, typename _Tp >`  
`const_mem_fun_ref_t< _Ret, _Tp > mem_fun_ref (_Ret(_Tp::* __f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const_mem_fun1_ref_t< _Ret, _Tp, _Arg > mem_fun_ref (_Ret(_Tp::* __f)(_Arg) const)`
- `void * memchr (void * __s, int __c, size_t __n)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵  
_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr const _Tp & min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr _Tp min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr _Filter min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr pair< _Tp, _Tp > minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< _Tp, _Tp > minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr pair< _Filter, _Filter > minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr pair< _Filter, _Filter > minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↵  
Iterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↵  
Iterator __last, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr pair< _Iter1, _Iter2 > mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↵  
_InputIterator2 __first2)`



- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > mismatch ( _InputIterator1 __first1, _InputIterator1 __last1, ↵`  
`_InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `float modf (float __x, float * __iptr)`
- `long double modf (long double __x, long double * __iptr)`
- `template<typename _Tp >`  
`constexpr std::remove_reference< _Tp >::type && move ( _Tp && __t) noexcept`
- `template<typename _II, typename _OI >`  
`constexpr _OI move ( _II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`constexpr _BI2 move_backward ( _BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Tp >`  
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type move_if_noexcept`  
`( _Tp & __x) noexcept`
- `template<typename _InputIterator >`  
`constexpr _InputIterator next ( _InputIterator __x, typename iterator_traits< _InputIterator >::difference_type ↵`  
`__n=1)`
- `template<typename _Blter >`  
`constexpr bool next_permutation ( _Blter, _Blter)`
- `template<typename _Blter, typename _Compare >`  
`constexpr bool next_permutation ( _Blter, _Blter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`constexpr bool next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `ios_base & noboolalpha (ios_base & __base)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool none_of ( _InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr bool none_of ( _Iter, _Iter, _Predicate)`
- `template<typename _Tp >`  
`_Tp constexpr norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`constexpr _Tp norm (const complex< _Tp > & __z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type norm ( _Tp __x)`
- `ios_base & noshowbase (ios_base & __base)`
- `ios_base & noshowpoint (ios_base & __base)`
- `ios_base & noshowpos (ios_base & __base)`
- `ios_base & noskipws (ios_base & __base)`
- `template<typename _Predicate >`  
`constexpr unary_negate< _Predicate > not1 (const _Predicate & __pred)`
- `template<typename _Predicate >`  
`constexpr binary_negate< _Predicate > not2 (const _Predicate & __pred)`
- `void notify_all_at_thread_exit (condition_variable &, unique_lock< mutex >)`
- `ios_base & nunitbuf (ios_base & __base)`

- `ios_base & nouppercase (ios_base & __base)`
- `template<typename _RAIter >  
constexpr void nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >  
constexpr void nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >  
constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >  
constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `ios_base & oct (ios_base & __base)`
- `template<typename _T1, typename _T2 >  
constexpr bool operator!= (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _StateT >  
bool operator!= (const fpos< _StateT > & __lhs, const fpos< _StateT > & __rhs)`
- `template<typename _CharT, typename _Traits >  
bool operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT, _Traits > & __b)`
- `template<typename _Tp, std::size_t _Nm >  
constexpr bool operator!= (const array< _Tp, _Nm > & __one, const array< _Tp, _Nm > & __two)`
- `bool operator!= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >  
bool operator!= (const stack< _Tp, _Seq > & __x, const stack< _Tp, _Seq > & __y)`
- `template<typename _Tp, typename _Seq >  
bool operator!= (const queue< _Tp, _Seq > & __x, const queue< _Tp, _Seq > & __y)`
- `template<typename _Iterator >  
constexpr bool operator!= (const reverse_iterator< _Iterator > & __x, const reverse_iterator< _Iterator > & __y)`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< struct std:: __not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std:: __not_equal_to, typename _Dom1::value_type >::result_type > operator!= (const _Expr< _Dom1, typename _Dom1::value_type > & __v, const _Expr< _Dom2, typename _Dom2::value_type > & __w)`
- `template<class _Dom >  
_Expr< _BinClos< struct std:: __not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std:: __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __v)`
- `template<class _Dom >  
_Expr< _BinClos< struct std:: __not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std:: __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > & __e, const valarray< typename _Dom::value_type > & __v)`
- `template<class _Dom >  
_Expr< _BinClos< struct std:: __not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std:: __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const valarray< typename _Dom::value_type > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<class _Dom >  
_Expr< _BinClos< struct std:: __not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std:: __not_equal_to, typename _Dom::value_type >::result_type > operator!= (const _Expr< _Dom, typename _Dom::value_type > & __v, const typename _Dom::value_type & __t)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m >  
bool operator!= (const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs, const std::linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs)`

- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator!= (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>`  
`bool operator!= (const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__lhs, const std::mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__rhs)`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator!= (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool operator!= (const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__lhs, const std::subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool operator!= (const std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__lhs, const std::discard\_block\_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > operator!= (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool operator!= (const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__lhs, const std::independent\_bits\_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator!= (const \_\_shared\_ptr< _Tp1, _Lp > &__a, const \_\_shared\_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (const \_\_shared\_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator!= (nullptr_t, const \_\_shared\_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator!= (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`

- `template<typename _Iterator >`  
`constexpr bool operator!= (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool operator!= (const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__lhs, const std::shuffle\_order\_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform\_int\_distribution< _IntType > &__d1, const std::uniform\_int\_distribution< _IntType > &__d2)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _IntType >`  
`bool operator!= (const std::uniform\_real\_distribution< _IntType > &__d1, const std::uniform\_real\_distribution< _IntType > &__d2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator!= (const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter, class _Alloc >`  
`bool operator!= (const match\_results< _Bi_iter, _Alloc > &__m1, const match\_results< _Bi_iter, _Alloc > &__m2)`
- `template<typename _RealType >`  
`bool operator!= (const std::normal\_distribution< _RealType > &__d1, const std::normal\_distribution< _RealType > &__d2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _RealType >`  
`bool operator!= (const std::lognormal\_distribution< _RealType > &__d1, const std::lognormal\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::gamma\_distribution< _RealType > &__d1, const std::gamma\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::chi\_squared\_distribution< _RealType > &__d1, const std::chi\_squared\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::cauchy\_distribution< _RealType > &__d1, const std::cauchy\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::fisher\_f\_distribution< _RealType > &__d1, const std::fisher\_f\_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool operator!= (const std::student\_t\_distribution< _RealType > &__d1, const std::student\_t\_distribution< _RealType > &__d2)`

- `bool operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >  
bool operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution< _IntType > &__d2)`
- `template<typename _IntType >  
bool operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _IntType > &__d2)`
- `template<typename _RealType >  
bool operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >  
bool operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >  
bool operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >  
bool operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
bool operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<class _Dom1, class _Dom2 >  
_Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__modulus, typename _Dom1::value_type >::result_type > operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >  
_Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >  
_Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >  
_Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__modulus, typename _Dom::value_type >::result_type > operator% (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename _fun< __modulus, _Tp >::`  
`result_type > operator% (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename _fun< __modulus, _Tp >::`  
`result_type > operator% (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename _fun< __modulus, _Tp >::result_`  
`type > operator% (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `constexpr _ios_Fmtflags operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr memory\_order operator& (memory\_order __m, __memory_order_modifier __mod)`
- `constexpr _ios_Openmode operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator& (launch __x, launch __y)`
- `constexpr _ios_ostate operator& (_ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::`  
`::__bitwise_and, typename _Dom1::value_type >::result_type > operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > operator& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `constexpr chars\_format operator& (chars\_format __lhs, chars\_format __rhs) noexcept`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, typename _fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, typename _fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, typename _fun< __bitwise_and, _Tp`  
`>::result_type > operator& (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const _`  
`Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const _Expr< _`  
`Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::`  
`::__logical_and, typename _Dom1::value_type >::result_type > operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > operator&& (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp >`  
`::result_type > operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > operator&& (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp`  
`>::result_type > operator&& (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `const _los_Fmtflags & operator&= (_los_Fmtflags &__a, _los_Fmtflags __b)`
- `const _los_Openmode & operator&= (_los_Openmode &__a, _los_Openmode __b)`
- `launch & operator&= (launch &__x, launch __y)`
- `const _los_losestate & operator&= (_los_losestate &__a, _los_losestate __b)`
- `constexpr chars\_format & operator&= (chars\_format &__lhs, chars\_format __rhs) noexcept`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::`  
`__multiplies, typename _Dom1::value_type >::result_type > operator* (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const typename _`  
`Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__multiplies, typename _Dom::value_type >::result_type > operator* (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >`  
`::result_type > operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`



- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > operator* (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > operator* (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > operator+ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > &__x)`
- `template<typename _Iterator >`  
`constexpr reverse_iterator< _Iterator > operator+ (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > operator+ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Iterator >`  
`constexpr move_iterator< _Iterator > operator+ (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`



- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const`  
`_CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc`  
`> &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _`  
`CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT`  
`__rhs)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const typename _Dom`  
`::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__minus,`  
`typename _Dom1::value_type >::result_type > operator- (const _Expr< _Dom1, typename _Dom1::value_type`  
`> &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _`  
`_fun< struct std::__minus, typename _Dom::value_type >::result_type > operator- (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator- (const complex< _Tp > &__x)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr auto operator- (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`  
`-> decltype(__y.base() - __x.base())`

- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> operator- (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> operator- (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __minus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __minus, _Tp >::result_type`  
`> operator- (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr auto operator- (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y) ->`  
`decltype(__x.base() - __y.base())`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`divides, typename _Dom1::value_type >::result_type > operator/ (const _Expr< _Dom1, typename _Dom1`  
`::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const typename _Dom`  
`::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > operator/ (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result_type`  
`> operator/ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __divides, _Tp >::result`  
`__type > operator/ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __divides, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __divides, _Tp >::result`  
`__type > operator/ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `bool operator< (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`constexpr bool operator< (const reverse\_iterator< _Iterator > &__x, const reverse\_iterator< _Iterator > &__y)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`less, typename _Dom1::value_type >::result_type > operator< (const _Expr< _Dom1, typename _Dom1::value`  
`__type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↵`  
`_fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const valarray< typename`  
`_Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _↵`  
`_fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const typename _Dom↵`  
`::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↵`  
`_fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↵`  
`_fun< struct std::__less, typename _Dom::value_type >::result_type > operator< (const _Expr< _Dom, type-`  
`name _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator< (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &↵`  
`__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less, _Tp >::result_type >`  
`operator< (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator< (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Iterator >`  
`constexpr bool operator< (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)`
- `template<class _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_time< _CharT > __f)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_left, typename _Dom1::value_type >::result_type > operator<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > operator<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _Ostream, typename _Tp >`  
`enable_if< __and< __not< is_lvalue_reference< _Ostream >, __is_convertible_to_basic_ostream< _Ostream >, __is_insertable< rvalue_ostream_type< _Ostream >, const _Tp & >::value, rvalue_ostream_type< _Ostream > >::type operator<< (_Ostream &&__os, const _Tp &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_type > operator<< (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_left, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_left, _Tp >::result_type > operator<< (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`

- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const  
std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
gamma_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const  
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const  
basic_string< _CharT, _Traits, _Alloc > &__str)`



- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator<= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`constexpr bool operator<= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _←`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`::__less_equal, typename _Dom1::value_type >::result_type > operator<= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const typename _←`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__less_equal, typename _Dom::value_type >::result_type > operator<= (const _Expr< _←`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _←`  
`_Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >←`  
`::result_type > operator<= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __less_equal, _Tp >←`  
`::result_type > operator<= (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __less_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __less_equal, _Tp >←`  
`::result_type > operator<= (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`

- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Iterator >`  
`constexpr bool operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator<= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _T1, typename _T2 >`  
`constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept`
- `template<typename _StateT >`  
`bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)`
- `template<typename _CharT, typename _Traits >`  
`bool operator== (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator== (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`constexpr bool operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__equal_to, typename _Dom1::value_type >::result_type > operator== (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`



- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__equal_to, typename _Dom::value_type >::result_type > operator== (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Res, typename... _Args>`  
`bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept`
- `template<typename _Res, typename... _Args>`  
`bool operator== (nullptr_t, const function< _Res(_Args...)> &__f) noexcept`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,`  
`_Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result<-`  
`type > operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >:-`  
`result_type > operator== (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >:-`  
`result_type > operator== (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc`  
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Iterator >`  
`constexpr bool operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value,`  
`_Hash, _Pred, _Alloc > &__y)`

- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _RealType >`  
`bool operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Bi_iter, typename _Alloc >`  
`bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT >`  
`__gnu_cxx::enable_if< __is_char< _CharT >::value, bool >::type operator== (const basic_string< _CharT > &__lhs, const basic_string< _CharT > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm >`  
`constexpr bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator> (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`constexpr bool operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > operator> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1 , class _Dom2 >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater, typename _Dom1::value_type >::result_type > operator> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL , typename _IteratorR >`  
`constexpr bool operator> (const reverse\_iterator< _IteratorL > &__x, const reverse\_iterator< _IteratorR > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Compare , typename _Alloc >`  
`bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > operator> (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator> (const forward\_list< _Tp, _Alloc > &__lx, const forward\_list< _Tp, _Alloc > &__ly)`
- `template<typename _IteratorL , typename _IteratorR >`  
`constexpr bool operator> (const move\_iterator< _IteratorL > &__x, const move\_iterator< _IteratorR > &__y)`
- `template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp1 , typename _Tp2 , _Lock_policy _Lp>`  
`bool operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp , _Lock_policy _Lp>`  
`bool operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp , _Lock_policy _Lp>`  
`bool operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Iterator >`  
`constexpr bool operator> (const move\_iterator< _Iterator > &__x, const move\_iterator< _Iterator > &__y)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)`
- `template<typename _Tp , typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator> (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `bool operator>= (thread::id __x, thread::id __y) noexcept`
- `template<typename _Tp, typename _Seq >`  
`bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)`
- `template<typename _Iterator >`  
`constexpr bool operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > operator>= (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`greater_equal, typename _Dom1::value_type >::result_type > operator>= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >`  
`&__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp`  
`>::result_type > operator>= (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp`  
`>::result_type > operator>= (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > operator>=` (const `valarray`< \_Tp > &\_\_v, const `valarray`< \_Tp > &\_\_w)
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool operator>=` (const `tuple`< \_TElements... > &\_\_t, const `tuple`< \_UElements... > &\_\_u)
- `template<typename _Tp, typename _Alloc >`  
`bool operator>=` (const `forward_list`< \_Tp, \_Alloc > &\_\_lx, const `forward_list`< \_Tp, \_Alloc > &\_\_ly)
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool operator>=` (const `move_iterator`< \_IteratorL > &\_\_x, const `move_iterator`< \_IteratorR > &\_\_y)
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool operator>=` (const `map`< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_x, const `map`< \_Key, \_Tp, \_Compare, \_Alloc > &\_\_y)
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>`  
`bool operator>=` (const `__shared_ptr`< \_Tp1, \_Lp > &\_\_a, const `__shared_ptr`< \_Tp2, \_Lp > &\_\_b) noexcept
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator>=` (const `__shared_ptr`< \_Tp, \_Lp > &\_\_a, `nullptr_t`) noexcept
- `template<typename _Tp, _Lock_policy _Lp>`  
`bool operator>=` (`nullptr_t`, const `__shared_ptr`< \_Tp, \_Lp > &\_\_a) noexcept
- `template<typename _Iterator >`  
`constexpr bool operator>=` (const `move_iterator`< \_Iterator > &\_\_x, const `move_iterator`< \_Iterator > &\_\_y)
- `template<typename _Tp, typename _Alloc >`  
`bool operator>=` (const `vector`< \_Tp, \_Alloc > &\_\_x, const `vector`< \_Tp, \_Alloc > &\_\_y)
- `template<typename _Tp, typename _Alloc >`  
`bool operator>=` (const `list`< \_Tp, \_Alloc > &\_\_x, const `list`< \_Tp, \_Alloc > &\_\_y)
- `template<typename _Tp, typename _Alloc >`  
`bool operator>=` (const `deque`< \_Tp, \_Alloc > &\_\_x, const `deque`< \_Tp, \_Alloc > &\_\_y)
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>=` (const `basic_string`< \_CharT, \_Traits, \_Alloc > &\_\_lhs, const `basic_string`< \_CharT, \_Traits, \_Alloc > &\_\_rhs) noexcept
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>=` (const `basic_string`< \_CharT, \_Traits, \_Alloc > &\_\_lhs, const `_CharT *`&\_\_rhs)
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool operator>=` (const `_CharT *`&\_\_lhs, const `basic_string`< \_CharT, \_Traits, \_Alloc > &\_\_rhs)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Resetiosflags` \_\_f)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Setiosflags` \_\_f)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Setbase` \_\_f)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Setfill`< \_CharT > \_\_f)
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream`< \_CharT, \_Traits > &\_\_is, `linear_congruential_engine`< \_UIntType, \_\_a, \_\_c, \_\_m > &\_\_lcr)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Setprecision` \_\_f)
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Setw` \_\_f)
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & operator>>` (`basic_istream`< \_CharT, \_Traits > &\_\_is, `_Get_money`< \_MoneyT > \_\_f)

- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_right, typename _Dom1::value_type >::result_type > operator>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > operator>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _CharT, typename _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, _Get_time< _CharT > > __f)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic\_istream< _CharT, _Traits > & operator>> (basic\_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, mersenne\_twister\_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, subtract\_with\_carry\_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, discard\_block\_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic\_istream< _CharT, _Traits > & operator>> (std::basic\_istream< _CharT, _Traits > &__is, shuffle\_order\_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _Istream, typename _Tp >`  
`enable\_if< __and< __not< is\_lvalue\_reference< _Istream > >, __is_convertible_to_basic_istream< _Istream >, __is_extractable< __rvalue_istream_type< _Istream >, _Tp && >::value, __rvalue_istream_type< _Istream > >::type operator>> (_Istream &&__is, _Tp &&__x)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > operator>> (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > operator>> (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`



- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
negative_binomial_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
poisson_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
binomial_distribution< _IntType > & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution<  
_IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
normal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution<  
_RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
lognormal_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
chi_squared_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
fisher_f_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
student_t_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
gamma_distribution< _RealType > & __x)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > & __is, __gnu_cxx::__versa_string<  
_CharT, _Traits, _Alloc, _Base > & __str)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
discrete_distribution< _IntType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
piecewise_constant_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
std::cauchy_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
piecewise_linear_distribution< _RealType > & __x)`
- `template<typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
std::bernoulli_distribution & __x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is,  
std::geometric_distribution< _IntType > & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _↵`  
`CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `constexpr _ios_Fmtflags operator^ ( _ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode operator^ ( _ios_Openmode __a, _ios_Openmode __b)`
- `constexpr launch operator^ (launch __x, launch __y)`
- `constexpr _ios_ostate operator^ ( _ios_ostate __a, _ios_ostate __b)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr< _↵`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > operator^ (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↵`  
`::__bitwise_xor, typename _Dom1::value_type >::result_type > operator^ (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `constexpr chars_format operator^ (chars_format __lhs, chars_format __rhs) noexcept`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >↵`  
`::result_type > operator^ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >↵`  
`::result_type > operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >↵`  
`::result_type > operator^ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `const _ios_Fmtflags & operator^= ( _ios_Fmtflags &__a, _ios_Fmtflags __b)`
- `const _ios_Openmode & operator^= ( _ios_Openmode &__a, _ios_Openmode __b)`
- `launch & operator^= (launch &__x, launch __y)`
- `const _ios_ostate & operator^= ( _ios_ostate &__a, _ios_ostate __b)`
- `constexpr chars_format & operator^= (chars_format &__lhs, chars_format __rhs) noexcept`



- constexpr `_los_Fmtflags operator|` (`_los_Fmtflags __a`, `_los_Fmtflags __b`)
- constexpr `memory_order operator|` (`memory_order __m`, `memory_order_modifier __mod`)
- constexpr `_los_Openmode operator|` (`_los_Openmode __a`, `_los_Openmode __b`)
- constexpr `launch operator|` (`launch __x`, `launch __y`)
- constexpr `_los_losestate operator|` (`_los_losestate __a`, `_los_losestate __b`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > operator|` (const `valarray< type-`  
`name _Dom::value_type > &__v`, const `_Expr< _Dom, typename _Dom::value_type > &__e`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > operator|` (const `_Expr< _Dom,`  
`typename _Dom::value_type > &__e`, const `valarray< typename _Dom::value_type > &__v`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > operator|` (const `_Expr< _Dom,`  
`typename _Dom::value_type > &__v`, const `typename _Dom::value_type &__t`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > operator|` (const `typename _Dom::value_type &__t`, const `_Expr< _Dom, typename _Dom::value_type > &__v`)
- template<class `_Dom1`, class `_Dom2` >  
`_Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__bitwise_or,`  
`typename _Dom1::value_type >::result_type > operator|` (const `_Expr< _Dom1, typename _Dom1::value_type > &__v`, const `_Expr< _Dom2, typename _Dom2::value_type > &__w`)
- constexpr `chars_format operator|` (`chars_format __lhs`, `chars_format __rhs`) noexcept
- template<typename `_Tp` >  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename _fun< __bitwise_or, _Tp >::result_type >`  
`operator|` (const `valarray< _Tp > &__v`, const `valarray< _Tp > &__w`)
- template<typename `_Tp` >  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename _fun< __bitwise_or, _Tp >::result_type >`  
`operator|` (const `typename valarray< _Tp >::value_type &__t`, const `valarray< _Tp > &__v`)
- template<typename `_Tp` >  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename _fun< __bitwise_or, _Tp >::result_type >`  
`operator|` (const `valarray< _Tp > &__v`, const `typename valarray< _Tp >::value_type &__t`)
- const `_los_Fmtflags & operator|=` (`_los_Fmtflags &__a`, `_los_Fmtflags __b`)
- const `_los_Openmode & operator|=` (`_los_Openmode &__a`, `_los_Openmode __b`)
- `launch & operator|=` (`launch &__x`, `launch __y`)
- const `_los_losestate & operator|=` (`_los_losestate &__a`, `_los_losestate __b`)
- constexpr `chars_format & operator|=` (`chars_format &__lhs`, `chars_format __rhs`) noexcept
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator||` (const `_Expr< _Dom,`  
`typename _Dom::value_type > &__e`, const `valarray< typename _Dom::value_type > &__v`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator||` (const `_Expr< _Dom,`  
`typename _Dom::value_type > &__v`, const `typename _Dom::value_type &__t`)
- template<class `_Dom` >  
`_Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator||` (const `typename _Dom::value_type &__t`, const `_Expr< _Dom, typename _Dom::value_type > &__v`)

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__logical_or, typename _Dom::value_type >::result_type > operator|| (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`__logical_or, typename _Dom1::value_type >::result_type > operator|| (const _Expr< _Dom1, typename _`  
`Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >::`  
`result_type > operator|| (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >::`  
`result_type > operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >::`  
`result_type > operator|| (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `constexpr _los_Fmtflags operator~ (_los_Fmtflags __a)`
- `constexpr _los_Openmode operator~ (_los_Openmode __a)`
- `constexpr launch operator~ (launch __x)`
- `constexpr _los_ostate operator~ (_los_ostate __a)`
- `constexpr chars\_format operator~ (chars\_format __fmt) noexcept`
- `template<typename _RAIter >`  
`constexpr void partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RandomAccessIterator >`  
`constexpr void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random`  
`AccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random`  
`AccessIterator __last, _Compare __comp)`
- `template<typename _Iter, typename _RAIter >`  
`constexpr _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`constexpr _RAIter partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`constexpr _RandomAccessIterator partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _Random`  
`AccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _Random`  
`AccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`constexpr _OutputIterator partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`constexpr _OutputIterator partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _`  
`BinaryOperation __binary_op)`
- `template<typename _BIter, typename _Predicate >`  
`constexpr _BIter partition (_BIter, _BIter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Iter, typename _Olter1, typename _Olter2, typename _Predicate >`  
`constexpr pair< _Olter1, _Olter2 > partition_copy (_Iter, _Iter, _Olter1, _Olter2, _Predicate)`

- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`constexpr pair< _OutputIterator1, _OutputIterator2 > partition\_copy (_InputIterator __first, _InputIterator __last,`  
`_OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`  
`constexpr _Filter partition\_point (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Tp >`  
`complex< _Tp > polar (const _Tp &, const _Tp &=0)`
- `template<typename _RandomAccessIterator >`  
`constexpr void pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void pop\_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`constexpr void pop\_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void pop\_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, int)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const _Tp &)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > pow (const _Tp &, const complex< _Tp > &)`
- `constexpr float pow (float __x, float __y)`
- `constexpr long double pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > pow (const typename valarray<`  
`_Tp >::value_type & __t, const valarray< _Tp > & __v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > & __e, const typename _Dom::value_type & __t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > pow (const _Expr< _Dom1, typename _Dom1::value_type > & __e1, const _Expr< _Dom2, typename _Dom2::value_type > & __e2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > & __v, const typename valarray< _Tp >::value_type & __t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const valarray< typename _Dom::valarray > & __v, const _Expr< _Dom, typename _Dom::value_type > & __e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > pow (const valarray< _Tp > & __v, const valarray< _Tp > & __w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > pow (const typename _Dom::value_type & __t, const _Expr< _Dom, typename _Dom::value_type > & __e)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const _Tp &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _BidirectionalIterator >`  
`constexpr _BidirectionalIterator prev (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)`
- `template<typename _BIter >`  
`constexpr bool prev_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`constexpr bool prev_permutation (_BIter, _BIter, _Compare)`
- `template<typename _BidirectionalIterator >`  
`constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`std::complex< _Tp > proj (const std::complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > proj (_Tp __x)`
- `template<typename _Arg, typename _Result >`  
`pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result >`  
`pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))`
- `template<typename _RandomAccessIterator >`  
`constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RAIter >`  
`constexpr void push_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void push_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT >`  
`_Put_time< _CharT > put_time (const std::tm *__tmb, const _CharT *__fmt)`
- `template<typename _CharT >`  
`auto quoted (const _CharT *__string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto quoted (const basic_string< _CharT, _Traits, _Alloc > &__string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto quoted (basic_string< _CharT, _Traits, _Alloc > &__string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`

- `template<typename _RAIter >`  
`void random_shuffle (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Generator >`  
`void random_shuffle (_RAIter, _RAIter, _Generator &&)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator && __rand)`
- `template<typename _Container >`  
`constexpr auto rbegin (_Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Container >`  
`constexpr auto rbegin (const _Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > rbegin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Tp >`  
`constexpr reverse_iterator< const _Tp * > rbegin (initializer_list< _Tp > __il) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp real (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type real (_Tp __x)`
- `template<typename _Filter, typename _Tp >`  
`constexpr _Filter remove (_Filter, _Filter, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`constexpr _OIter remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`constexpr _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`constexpr _OIter remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`constexpr _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _Filter, typename _Predicate >`  
`constexpr _Filter remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _Container >`  
`constexpr auto rend (_Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Container >`  
`constexpr auto rend (const _Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > rend (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Tp >`  
`constexpr reverse_iterator< const _Tp * > rend (initializer_list< _Tp > __il) noexcept`
- `template<typename _Filter, typename _Tp >`  
`constexpr void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`constexpr _OIter replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`

- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`constexpr _OutputIterator replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`constexpr _OIter replace\_copy\_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`constexpr _OutputIterator replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`constexpr void replace\_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`constexpr void replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__`  
`new_value)`
- `_Resetiosflags resetiosflags (ios_base::fmtflags __mask)`
- `void rethrow\_exception (exception_ptr)`
- `template<typename _Ex >`  
`void rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Tp >`  
`void return\_temporary\_buffer (_Tp *__p)`
- `template<typename _BIter >`  
`constexpr void reverse (_BIter, _BIter)`
- `template<typename _BidirectionalIterator >`  
`constexpr void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BIter, typename _OIter >`  
`constexpr _OIter reverse\_copy (_BIter, _BIter, _OIter)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`constexpr _OutputIterator reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _Output`  
`Iterator __result)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type riemann\_zeta (_Tp __s)`
- `float riemann\_zetaf (float __s)`
- `long double riemann\_zetal (long double __s)`
- `ios_base & right (ios_base &__base)`
- `template<typename _Filter, typename _OIter >`  
`constexpr _OIter rotate\_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _ForwardIterator, typename _OutputIterator >`  
`constexpr _OutputIterator rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __`  
`__last, _OutputIterator __result)`
- `ios_base & scientific (ios_base &__base)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr _Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __`  
`__first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __`  
`__first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`  
`constexpr _Filter search\_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`constexpr _Filter search\_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`

- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`constexpr _ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const`  
`_Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`constexpr _ForwardIterator search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const`  
`_Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter set\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter set\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter set\_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter set\_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __↵`  
`first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __↵`  
`first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `new\_handler set\_new\_handler (new\_handler) throw ()`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter set\_symmetric\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter set\_symmetric\_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _Input↵`  
`Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `terminate\_handler set\_terminate (terminate\_handler) noexcept`
- `unexpected\_handler set\_unexpected (unexpected\_handler) noexcept`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter set\_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↵`  
`_InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `_Setbase setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > setfill (_CharT __c)`
- `_Setiosflags setiosflags (ios_base::fmtflags __mask)`



- `_Setprecision` [setprecision](#) (int \_\_n)
- `_Setw` [setw](#) (int \_\_n)
- `ios_base` & `showbase` (`ios_base` & \_\_base)
- `ios_base` & `showpoint` (`ios_base` & \_\_base)
- `ios_base` & `showpos` (`ios_base` & \_\_base)
- `template<typename _RAIter, typename _UGenerator >`  
`void` **shuffle** (`_RAIter`, `_RAIter`, `_UGenerator` &&)
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void` **shuffle** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_UniformRandomNumberGenerator` && \_\_g)
- `template<typename _Tp >`  
`complex<_Tp>` **sin** (const `complex<_Tp>` &)
- `constexpr float` **sin** (float \_\_x)
- `constexpr long double` **sin** (long double \_\_x)
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type` **sin** (`_Tp` \_\_x)
- `template<class _Dom >`  
`_Expr<_UnClos<struct std::_Sin, _Expr, _Dom>, typename _Dom::value_type>` **sin** (const `_Expr<_Dom, typename _Dom::value_type>` & \_\_e)
- `template<typename _Tp >`  
`_Expr<_UnClos<struct std::_Sin, _ValArray, _Tp>, _Tp>` **sin** (const `valarray<_Tp>` & \_\_v)
- `template<typename _Tp >`  
`complex<_Tp>` **sinh** (const `complex<_Tp>` &)
- `constexpr float` **sinh** (float \_\_x)
- `constexpr long double` **sinh** (long double \_\_x)
- `template<class _Dom >`  
`_Expr<_UnClos<struct std::_Sinh, _Expr, _Dom>, typename _Dom::value_type>` **sinh** (const `_Expr<_Dom, typename _Dom::value_type>` & \_\_e)
- `template<typename _Tp >`  
`_Expr<_UnClos<struct std::_Sinh, _ValArray, _Tp>, _Tp>` **sinh** (const `valarray<_Tp>` & \_\_v)
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<__is_integer<_Tp>::__value, double>::__type` **sinh** (`_Tp` \_\_x)
- `ios_base` & `skipws` (`ios_base` & \_\_base)
- `template<typename _RAIter >`  
`constexpr void` **sort** (`_RAIter`, `_RAIter`)
- `template<typename _RAIter, typename _Compare >`  
`constexpr void` **sort** (`_RAIter`, `_RAIter`, `_Compare`)
- `template<typename _RandomAccessIterator >`  
`constexpr void` **sort** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void` **sort** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Compare` \_\_comp)
- `template<typename _RandomAccessIterator >`  
`constexpr void` **sort\_heap** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last)
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void` **sort\_heap** (`_RandomAccessIterator` \_\_first, `_RandomAccessIterator` \_\_last, `_Compare` \_\_comp)
- `template<typename _RAIter >`  
`constexpr void` **sort\_heap** (`_RAIter`, `_RAIter`)
- `template<typename _RAIter, typename _Compare >`  
`constexpr void` **sort\_heap** (`_RAIter`, `_RAIter`, `_Compare`)
- `template<typename _Tp >`  
`__gnu_cxx::__promote<_Tp>::__type` **sph\_bessel** (unsigned int \_\_n, `_Tp` \_\_x)
- `float` **sph\_besself** (unsigned int \_\_n, float \_\_x)



- long double [sph\\_bessell](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [sph\\_legendre](#) (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float [sph\\_legendref](#) (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double [sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
[complex](#)< \_Tp > [sqrt](#) (const [complex](#)< \_Tp > &)
- template<typename \_Tp >  
\_Expr< \_UnClos< struct std::\_Sqrt, \_ValArray, \_Tp >, \_Tp > [sqrt](#) (const [valarray](#)< \_Tp > &\_\_v)
- template<class \_Dom >  
\_Expr< \_UnClos< struct std::\_Sqrt, \_Expr, \_Dom >, typename \_Dom::value\_type > [sqrt](#) (const \_Expr< \_Dom, typename \_Dom::value\_type > &\_\_e)
- constexpr float [sqrt](#) (float \_\_x)
- constexpr long double [sqrt](#) (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type [sqrt](#) (\_Tp \_\_x)
- template<typename \_Blter, typename \_Predicate >  
\_Blter [stable\\_partition](#) (\_Blter, \_Blter, \_Predicate)
- template<typename \_ForwardIterator, typename \_Predicate >  
\_ForwardIterator [stable\\_partition](#) (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Predicate \_\_pred)
- template<typename \_RAIter >  
void [stable\\_sort](#) (\_RAIter, \_RAIter)
- template<typename \_RAIter, typename \_Compare >  
void [stable\\_sort](#) (\_RAIter, \_RAIter, \_Compare)
- template<typename \_RandomAccessIterator >  
void [stable\\_sort](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last)
- template<typename \_RandomAccessIterator, typename \_Compare >  
void [stable\\_sort](#) (\_RandomAccessIterator \_\_first, \_RandomAccessIterator \_\_last, \_Compare \_\_comp)
- template<typename \_Tp, typename \_Tp1, \_Lock\_policy \_Lp>  
\_\_shared\_ptr< \_Tp, \_Lp > [static\\_pointer\\_cast](#) (const \_\_shared\_ptr< \_Tp1, \_Lp > &\_\_r) noexcept
- char \* [strchr](#) (char \*\_\_s, int \_\_n)
- char \* [strpbrk](#) (char \*\_\_s1, const char \*\_\_s2)
- char \* [strrchr](#) (char \*\_\_s, int \_\_n)
- char \* [strstr](#) (char \*\_\_s1, const char \*\_\_s2)
- void [swap](#) (\_Bit\_reference \_\_x, \_Bit\_reference \_\_y) noexcept
- void [swap](#) (\_Bit\_reference \_\_x, bool &\_\_y) noexcept
- void [swap](#) (bool &\_\_x, \_Bit\_reference \_\_y) noexcept
- template<typename \_Tp >  
constexpr [enable\\_if](#)< \_\_and< \_\_not< \_\_is\_tuple\_like< \_Tp >, [is\\_move\\_constructible](#)< \_Tp >, [is\\_move\\_assignable](#)< \_Tp >::\_\_value >::type [swap](#) (\_Tp &\_\_a, \_Tp &\_\_b) noexcept(*/\*conditional \*/*)  
[is\\_nothrow\\_move\\_assignable](#)< \_Tp >>
- template<typename \_Tp, size\_t \_Nm>  
constexpr [enable\\_if](#)< \_\_is\_swappable< \_Tp >::\_\_value >::type [swap](#) (\_Tp(&\_\_a)[\_Nm], \_Tp(&\_\_b)[\_Nm])  
noexcept(*/\*conditional \*/*)
- void [swap](#) (thread &\_\_x, thread &\_\_y) noexcept
- template<typename \_Tp, std::size\_t \_Nm>  
constexpr [enable\\_if](#)< ::\_\_array\_traits< \_Tp, \_Nm >::\_\_is\_swappable::value >::type [swap](#) ([array](#)< \_Tp, \_Nm > &\_\_one, [array](#)< \_Tp, \_Nm > &\_\_two) noexcept(noexcept(\_\_one.swap(\_\_two)))

- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< !::__array_traits< _Tp, _Nm >::is_swappable::value >::type swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type swap (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Res, typename... _Args>`  
`void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept`
- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`enable_if< __and< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type swap (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_istream< _CharT, _Traits, _Allocator > &__x, basic_istream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >`  
`void swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Res >`  
`void swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)`
- `template<class _CharT, class _Traits >`  
`void swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

- `template<typename _Tp, _Lock_policy _Lp>`  
`void swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Res, typename... _ArgTypes>`  
`void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`constexpr enable_if< !__and< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`void swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, _Lock_policy _Lp>`  
`void swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Bi_iter, typename _Alloc >`  
`void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)`
- `template<typename _Tp >`  
`constexpr Require< __not< __is_tuple_like< _Tp >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > swap (_Tp &, _Tp &) noexcept(__and< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr __enable_if_t< __is_swappable< _Tp >::value > swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr _Filter2 swap_ranges (_Filter1, _Filter1, _Filter2)`
- `template<typename _Tp >`  
`complex< _Tp > tan (const complex< _Tp > &)`

- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > tan (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > tan (const valarray< _Tp > &__v)`
- `constexpr float tan (float __x)`
- `constexpr long double tan (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tan (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > tanh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > tanh (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > tanh (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e)`
- `constexpr float tanh (float __x)`
- `constexpr long double tanh (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type tanh (_Tp __x)`
- `void terminate () noexcept`
- `template<typename _Tp >`  
`void throw\_with\_nested (_Tp &&__t)`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > tie (_Elements &... __args) noexcept`
- `to\_chars\_result to\_chars (char *__first, char *__last, char __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, signed char __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, unsigned char __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, signed short __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, unsigned short __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, signed int __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, unsigned int __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, signed long __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, unsigned long __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, signed long long __value, int __base=10)`
- `to\_chars\_result to\_chars (char *__first, char *__last, unsigned long long __value, int __base=10)`
- `to\_chars\_result to\_chars (char *, char *, bool, int=10)=delete`
- `string to\_string (int __val)`
- `string to\_string (unsigned __val)`
- `string to\_string (long __val)`
- `string to\_string (unsigned long __val)`
- `string to\_string (long long __val)`
- `string to\_string (unsigned long long __val)`
- `template<typename _CharT >`  
`_CharT tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT toupper (_CharT __c, const locale &__loc)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`constexpr _OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`  
`constexpr _OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`constexpr _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`constexpr _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`  
`int try_lock (_Lock1 & __l1, _Lock2 & __l2, _Lock3 &... __l3)`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::type`
- `bool uncaught_exception () noexcept`
- `int uncaught_exceptions () noexcept`
- `void undeclare_no_pointers (char *, size_t)`
- `template<typename _Tp >`  
`_Tp * undeclare_reachable (_Tp * __p)`
- `void unexpected ()`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`_ForwardIterator uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp & __x)`
- `template<typename _Filter >`  
`constexpr _Filter unique (_Filter, _Filter)`
- `template<typename _Filter, typename _BinaryPredicate >`  
`constexpr _Filter unique (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, typename _OIter >`  
`constexpr _OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`  
`constexpr _OIter unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)`
- `ios_base & unitbuf (ios_base & __base)`
- `template<typename _Filter, typename _Tp >`  
`constexpr _Filter upper_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr _Filter upper_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __val, _Compare __comp)`

- `ios_base & uppercase (ios_base & __base)`
  - `template<typename _Facet >`  
`const _Facet & use_facet (const locale & __loc)`
  - `wchar_t * wcschr (wchar_t * __p, wchar_t __c)`
  - `wchar_t * wcsbrk (wchar_t * __s1, const wchar_t * __s2)`
  - `wchar_t * wcsrchr (wchar_t * __p, wchar_t __c)`
  - `wchar_t * wcsstr (wchar_t * __s1, const wchar_t * __s2)`
  - `wchar_t * wmemchr (wchar_t * __p, wchar_t __c, size_t __n)`
  - `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > & __is)`
- 
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
  - `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
  - `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > & __x, const bitset< _Nb > & __y) noexcept`
- 
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > & __is, bitset< _Nb > & __x)`
  - `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > & __os, const bitset< _Nb > & __x)`
- 
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > & __x, const complex< _Tp > & __y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > operator+ (const complex< _Tp > & __x, const _Tp & __y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > operator+ (const _Tp & __x, const complex< _Tp > & __y)`
- 
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator- (const complex< _Tp > & __x, const complex< _Tp > & __y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > operator- (const complex< _Tp > & __x, const _Tp & __y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > operator- (const _Tp & __x, const complex< _Tp > & __y)`

- `template<typename _Tp >`  
`constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool operator!= (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)`

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`
  - `template<>`  
`basic_istream< char > & operator>> (basic_istream< char > &__in, char *__s)`
  - `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char *__s)`
  - `template<class _Traits >`  
`basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char *__s)`
- 
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)`
  - `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
- 
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
  - `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
  - `template<typename _Traits >`  
`basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

## Matching, Searching, and Replacing

- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Alloc, typename _Rx_traits >`  
`bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)`



- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_`  
`_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Ch_type, class _Rx_traits >`  
`bool regex_match (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex<`  
`_Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`  
`bool regex_search (const _Ch_type * __s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type`  
`__f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex<`  
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_`  
`_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename`  
`basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type,`  
`_Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _`  
`_Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__`  
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type * __fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type > regex_replace (const _Ch_type * __s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`

- `template<typename _Rx_traits, typename _Ch_type >`  
`basic_string<_Ch_type> regex_replace (const _Ch_type * __s, const basic_regex<_Ch_type, _Rx_traits>`  
`& __e, const _Ch_type * __fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)`

## Variables

- `static ios_base::Init __ioinit`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr bool __is_nothrow_uses_allocator_constructible_v`
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`constexpr bool __is_uses_allocator_constructible_v`
- `std::is_reference GLIBCXX_DEPRECATED SUGGEST`
- `constexpr adopt_lock_t adopt_lock`
- `constexpr allocator_arg_t allocator_arg`
- `constexpr defer_lock_t defer_lock`
- `constexpr _Swallow_assign ignore`
- `template<typename _Tp >`  
`constexpr bool is_nothrow_swappable_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool is_nothrow_swappable_with_v`
- `template<typename _Tp >`  
`constexpr bool is_swappable_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool is_swappable_with_v`
- `error_code make_error_code (errc) noexcept`
- `const nothrow_t nothrow`
- `decltype(nullptr) typedef nullptr_t`
- `constexpr piecewise_construct_t piecewise_construct`
- `constexpr try_to_lock_t try_to_lock`

## Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- `istream cin`
- `ostream cout`
- `ostream cerr`
- `ostream clog`
- `wistream wcin`
- `wostream wcout`
- `wostream wcerr`
- `wostream wclog`

### 3.10.1 Detailed Description

ISO C++ entities toplevel namespace is std.

ISO C++ inline namespace for literal suffixes.

### 3.10.2 Typedef Documentation

#### 3.10.2.1 \_\_ptr\_rebind

```
template<typename _Ptr , typename _Tp >
using std::__ptr_rebind = typedef typename pointer_traits<_Ptr>::template rebind<_Tp>
```

Convenience alias for rebinding pointers.

Definition at line 152 of file ptr\_traits.h.

#### 3.10.2.2 \_\_umap\_traits

```
template<bool _Cache>
using std::__umap_traits = typedef __detail::_Hashtable_traits<_Cache, false, true>
```

Base types for unordered\_map.

Definition at line 40 of file unordered\_map.h.

#### 3.10.2.3 \_\_ummap\_traits

```
template<bool _Cache>
using std::__ummap_traits = typedef __detail::_Hashtable_traits<_Cache, false, false>
```

Base types for unordered\_multimap.

Definition at line 57 of file unordered\_map.h.

#### 3.10.2.4 \_\_umset\_traits

```
template<bool _Cache>
using std::__umset_traits = typedef __detail::_Hashtable_traits<_Cache, true, false>
```

Base types for unordered\_multiset.

Definition at line 55 of file unordered\_set.h.

#### 3.10.2.5 \_\_uset\_traits

```
template<bool _Cache>
using std::__uset_traits = typedef __detail::__Hashtable_traits<_Cache, true, true>
```

Base types for unordered\_set.

Definition at line 40 of file unordered\_set.h.

#### 3.10.2.6 index\_sequence

```
template<size_t... _Idx>
using std::index_sequence = typedef integer_sequence<size_t, _Idx...>
```

Alias template index\_sequence.

Definition at line 345 of file utility.

#### 3.10.2.7 index\_sequence\_for

```
template<typename... _Types>
using std::index_sequence_for = typedef make_index_sequence<sizeof...(_Types)>
```

Alias template index\_sequence\_for.

Definition at line 353 of file utility.

#### 3.10.2.8 make\_index\_sequence

```
template<size_t _Num>
using std::make_index_sequence = typedef make_integer_sequence<size_t, _Num>
```

Alias template make\_index\_sequence.

Definition at line 349 of file utility.

#### 3.10.2.9 make\_integer\_sequence

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence = typedef integer_sequence<_Tp, __integer_pack(_Num)...>
```

Alias template make\_integer\_sequence.

Definition at line 338 of file utility.

#### 3.10.2.10 new\_handler

```
typedef void(* std::new_handler) ()
```

If you write your own error handler to be called by `new`, it must be of this type.

Definition at line 103 of file `new`.

#### 3.10.2.11 streamoff

```
typedef long long std::streamoff
```

Type used by `fpos`, `char_traits<char>`, and `char_traits<wchar_t>`.

In clauses 21.1.3.1 and 27.4.1 `streamoff` is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, `streamoff` was typedef `long`.

Definition at line 94 of file `postypes.h`.

#### 3.10.2.12 streampos

```
typedef fpos<mbstate_t> std::streampos
```

File position for char streams.

Definition at line 234 of file `postypes.h`.

#### 3.10.2.13 streamsize

```
typedef ptrdiff_t std::streamsize
```

Integral type for I/O operation counts and buffer sizes.

Definition at line 98 of file `postypes.h`.

#### 3.10.2.14 u16streampos

```
typedef fpos<mbstate_t> std::u16streampos
```

File position for `char16_t` streams.

Definition at line 245 of file `postypes.h`.

### 3.10.2.15 u32streampos

```
typedef fpos<mbstate_t> std::u32streampos
```

File position for char32\_t streams.

Definition at line 247 of file postypes.h.

### 3.10.2.16 wstreampos

```
typedef fpos<mbstate_t> std::wstreampos
```

File position for wchar\_t streams.

Definition at line 236 of file postypes.h.

## 3.10.3 Enumeration Type Documentation

### 3.10.3.1 anonymous enum

```
anonymous enum
```

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html) This controls some aspect of the sort routines.

Definition at line 1880 of file stl\_algo.h.

### 3.10.3.2 chars\_format

```
enum std::chars_format [strong]
```

floating-point format for primitive numerical conversion

Definition at line 658 of file charconv.

### 3.10.3.3 float\_denorm\_style

```
enum std::float_denorm_style
```

Describes the denormalization for floating-point types.

These values represent the presence or absence of a variable number of exponent bits. This type is used in the std↵::numeric\_limits class.

**Enumerator**

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| denorm_indeterminate | Indeterminate at compile time whether denormalized values are allowed. |
| denorm_absent        | The type does not allow denormalized values.                           |
| denorm_present       | The type allows denormalized values.                                   |

Definition at line 182 of file limits.

**3.10.3.4 float\_round\_style**

```
enum std::float_round_style
```

Describes the rounding style for floating-point types.

This is used in the std::numeric\_limits class.

**Enumerator**

|                           |                                     |
|---------------------------|-------------------------------------|
| round_toward_zero         | Intermediate.                       |
| round_to_nearest          | To zero.                            |
| round_toward_infinity     | To the nearest representable value. |
| round_toward_neg_infinity | To infinity.                        |

Definition at line 167 of file limits.

**3.10.3.5 io\_errc**

```
enum std::io_errc [strong]
```

I/O error code.

Definition at line 203 of file ios\_base.h.

**3.10.4 Function Documentation****3.10.4.1 \_\_allocate\_guarded()**

```
template<typename _Alloc >
__allocated_ptr<_Alloc> std::__allocate_guarded (
 _Alloc & __a)
```

Allocate space for a single object using \_\_a.

Definition at line 95 of file allocated\_ptr.h.

References std::allocator\_traits< \_Alloc >::allocate().

#### 3.10.4.2 `__final_insertion_sort()`

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::__final_insertion_sort (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Compare __comp)
```

This is a helper function for the sort routine.

Definition at line 1886 of file `stl_algo.h`.

References `__insertion_sort()`, and `__unguarded_insertion_sort()`.

#### 3.10.4.3 `__find_if()` [1/2]

```
template<typename _InputIterator , typename _Predicate >
constexpr _InputIterator std::__find_if (
 _InputIterator __first,
 _InputIterator __last,
 _Predicate __pred,
 input_iterator_tag) [inline]
```

This is an overload used by find algos for the Input Iterator case.

Definition at line 1909 of file `stl_algobase.h`.

Referenced by `__find_if_not()`, and `__search_n_aux()`.

#### 3.10.4.4 `__find_if()` [2/2]

```
template<typename _RandomAccessIterator , typename _Predicate >
constexpr _RandomAccessIterator std::__find_if (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Predicate __pred,
 random_access_iterator_tag)
```

This is an overload used by find algos for the RAI case.

Definition at line 1921 of file `stl_algobase.h`.



#### 3.10.4.5 \_\_find\_if\_not()

```
template<typename _InputIterator , typename _Predicate >
constexpr _InputIterator std::__find_if_not (
 _InputIterator __first,
 _InputIterator __last,
 _Predicate __pred) [inline]
```

Provided for `stable_partition` to use.

Definition at line 103 of file `stl_algo.h`.

References `__find_if()`.

#### 3.10.4.6 \_\_find\_if\_not\_n()

```
template<typename _InputIterator , typename _Predicate , typename _Distance >
constexpr _InputIterator std::__find_if_not_n (
 _InputIterator __first,
 _Distance & __len,
 _Predicate __pred)
```

Like `find_if_not()`, but uses and updates a count of the remaining range length instead of comparing against an end iterator.

Definition at line 117 of file `stl_algo.h`.

#### 3.10.4.7 \_\_gcd()

```
template<typename _EuclideanRingElement >
constexpr _EuclideanRingElement std::__gcd (
 _EuclideanRingElement __m,
 _EuclideanRingElement __n)
```

This is a helper function for the rotate algorithm specialized on RAs. It returns the greatest common divisor of two integer values.

Definition at line 1224 of file `stl_algo.h`.

#### 3.10.4.8 \_\_gen\_two\_uniform\_ints()

```
template<typename _IntType , typename _UniformRandomBitGenerator >
pair<_IntType, _IntType> std::__gen_two_uniform_ints (
 _IntType __b0,
 _IntType __b1,
 _UniformRandomBitGenerator && __g)
```

Generate two uniformly distributed integers using a single distribution invocation.

**Parameters**

|                   |                                         |
|-------------------|-----------------------------------------|
| <code>__b0</code> | The upper bound for the first integer.  |
| <code>__b1</code> | The upper bound for the second integer. |
| <code>__g</code>  | A UniformRandomBitGenerator.            |

**Returns**

A pair (i, j) with i and j uniformly distributed over [0, \_\_b0) and [0, \_\_b1), respectively.

Requires: `__b0 * __b1 <= __g.max() - __g.min()`.

Using `uniform_int_distribution` with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since `uniform_int_distribution` does not store leftover randomness between invocations.

If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.

Definition at line 3730 of file `stl_algo.h`.

Referenced by `__sample()`.

**3.10.4.9 `__heap_select()`**

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::__heap_select (
 _RandomAccessIterator __first,
 _RandomAccessIterator __middle,
 _RandomAccessIterator __last,
 _Compare __comp)
```

This is a helper function for the sort routines.

Definition at line 1667 of file `stl_algo.h`.

**3.10.4.10 `__inplace_stable_sort()`**

```
template<typename _RandomAccessIterator , typename _Compare >
void std::__inplace_stable_sort (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Compare __comp)
```

This is a helper function for the stable sorting routines.

Definition at line 2778 of file `stl_algo.h`.

References `__insertion_sort()`, and `__merge_without_buffer()`.

#### 3.10.4.11 \_\_insertion\_sort()

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::__insertion_sort (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Compare __comp)
```

This is a helper function for the sort routine.

Definition at line 1844 of file `stl_algo.h`.

Referenced by `__final_insertion_sort()`, and `__inplace_stable_sort()`.

#### 3.10.4.12 \_\_introsort\_loop()

```
template<typename _RandomAccessIterator , typename _Size , typename _Compare >
constexpr void std::__introsort_loop (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Size __depth_limit,
 _Compare __comp)
```

This is a helper function for the sort routine.

Definition at line 1950 of file `stl_algo.h`.

References `__unguarded_partition_pivot()`.

#### 3.10.4.13 \_\_lg()

```
constexpr int std::__lg (
 int __n) [inline]
```

This is a helper function for the sort routines and for `random.tcc`.

Definition at line 1364 of file `stl_algobase.h`.

Referenced by `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator>()()`, and `std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed()`.

#### 3.10.4.14 `__merge_adaptive()`

```
template<typename _BidirectionalIterator , typename _Distance , typename _Pointer , typename _Compare >
void std::__merge_adaptive (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 _Distance __len1,
 _Distance __len2,
 _Pointer __buffer,
 _Distance __buffer_size,
 _Compare __comp)
```

This is a helper function for the merge routines.

Definition at line 2432 of file `stl_algo.h`.

#### 3.10.4.15 `__merge_without_buffer()`

```
template<typename _BidirectionalIterator , typename _Distance , typename _Compare >
void std::__merge_without_buffer (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 _Distance __len1,
 _Distance __len2,
 _Compare __comp)
```

This is a helper function for the merge routines.

Definition at line 2493 of file `stl_algo.h`.

References `advance()`, and `iter_swap()`.

Referenced by `__inplace_stable_sort()`.

#### 3.10.4.16 `__move_median_to_first()`

```
template<typename _Iterator , typename _Compare >
constexpr void std::__move_median_to_first (
 _Iterator __result,
 _Iterator __a,
 _Iterator __b,
 _Iterator __c,
 _Compare __comp)
```

Swaps the median value of `*__a`, `*__b` and `*__c` under `__comp` to `*__result`.

Definition at line 79 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

#### 3.10.4.17 \_\_move\_merge()

```
template<typename _InputIterator , typename _OutputIterator , typename _Compare >
_OutputIterator std::__move_merge (
 _InputIterator __first1,
 _InputIterator __last1,
 _InputIterator __first2,
 _InputIterator __last2,
 _OutputIterator __result,
 _Compare __comp)
```

This is a helper function for the \_\_merge\_sort\_loop routines.

Definition at line 2655 of file stl\_algo.h.

#### 3.10.4.18 \_\_move\_merge\_adaptive()

```
template<typename _InputIterator1 , typename _InputIterator2 , typename _OutputIterator , typename
_Compare >
void std::__move_merge_adaptive (
 _InputIterator1 __first1,
 _InputIterator1 __last1,
 _InputIterator2 __first2,
 _InputIterator2 __last2,
 _OutputIterator __result,
 _Compare __comp)
```

This is a helper function for the \_\_merge\_adaptive routines.

Definition at line 2325 of file stl\_algo.h.

#### 3.10.4.19 \_\_move\_merge\_adaptive\_backward()

```
template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Bidirectional↵
Iterator3 , typename _Compare >
void std::__move_merge_adaptive_backward (
 _BidirectionalIterator1 __first1,
 _BidirectionalIterator1 __last1,
 _BidirectionalIterator2 __first2,
 _BidirectionalIterator2 __last2,
 _BidirectionalIterator3 __result,
 _Compare __comp)
```

This is a helper function for the \_\_merge\_adaptive routines.

Definition at line 2351 of file stl\_algo.h.

**3.10.4.20** `__partition()` [1/2]

```
template<typename _ForwardIterator , typename _Predicate >
constexpr _ForwardIterator std::__partition (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Predicate __pred,
 forward_iterator_tag)
```

This is a helper function...

Definition at line 1486 of file `stl_algo.h`.

References `iter_swap()`.

**3.10.4.21** `__partition()` [2/2]

```
template<typename _BidirectionalIterator , typename _Predicate >
constexpr _BidirectionalIterator std::__partition (
 _BidirectionalIterator __first,
 _BidirectionalIterator __last,
 _Predicate __pred,
 bidirectional_iterator_tag)
```

This is a helper function...

Definition at line 1512 of file `stl_algo.h`.

References `iter_swap()`.

**3.10.4.22** `__reverse()` [1/2]

```
template<typename _BidirectionalIterator >
constexpr void std::__reverse (
 _BidirectionalIterator __first,
 _BidirectionalIterator __last,
 bidirectional_iterator_tag)
```

This is an uglified `reverse(_BidirectionalIterator, _BidirectionalIterator)` overloaded for bidirectional iterators.

Definition at line 1120 of file `stl_algo.h`.

References `iter_swap()`.

**3.10.4.23 \_\_reverse()** [2/2]

```
template<typename _RandomAccessIterator >
constexpr void std::__reverse (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 random_access_iterator_tag)
```

This is an uglified reverse(\_BidirectionalIterator, \_BidirectionalIterator) overloaded for random access iterators.

Definition at line 1141 of file stl\_algo.h.

References iter\_swap().

**3.10.4.24 \_\_rotate\_adaptive()**

```
template<typename _BidirectionalIterator1 , typename _BidirectionalIterator2 , typename _Distance
>
_BidirectionalIterator1 std::__rotate_adaptive (
 _BidirectionalIterator1 __first,
 _BidirectionalIterator1 __middle,
 _BidirectionalIterator1 __last,
 _Distance __len1,
 _Distance __len2,
 _BidirectionalIterator2 __buffer,
 _Distance __buffer_size)
```

This is a helper function for the merge routines.

Definition at line 2394 of file stl\_algo.h.

**3.10.4.25 \_\_sample()** [1/2]

```
template<typename _InputIterator , typename _RandomAccessIterator , typename _Size , typename _URBG
UniformRandomBitGenerator >
_RandomAccessIterator std::__sample (
 _InputIterator __first,
 _InputIterator __last,
 input_iterator_tag ,
 _RandomAccessIterator __out,
 random_access_iterator_tag ,
 _Size __n,
 UniformRandomBitGenerator && __g)
```

Reservoir sampling algorithm.

Definition at line 5767 of file stl\_algo.h.

Referenced by \_\_gnu\_parallel::multiseq\_selection().

**3.10.4.26** `__sample()` [2/2]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _Cat , typename _Size ,
typename _UniformRandomBitGenerator >
_OutputIterator std::__sample (
 _ForwardIterator __first,
 _ForwardIterator __last,
 forward_iterator_tag ,
 _OutputIterator __out,
 _Cat ,
 _Size __n,
 _UniformRandomBitGenerator && __g)
```

Selection sampling algorithm.

Definition at line 5794 of file `stl_algo.h`.

References `__gen_two_uniform_ints()`, `distance()`, `std::pair< _T1, _T2 >::first`, `min()`, and `std::pair< _T1, _T2 >::second`.

**3.10.4.27** `__search_n_aux()` [1/2]

```
template<typename _ForwardIterator , typename _Integer , typename _UnaryPredicate >
constexpr _ForwardIterator std::__search_n_aux (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Integer __count,
 _UnaryPredicate __unary_pred,
 std::forward_iterator_tag)
```

This is an helper function for `search_n` overloaded for forward iterators.

Definition at line 194 of file `stl_algo.h`.

References `__find_if()`.

**3.10.4.28** `__search_n_aux()` [2/2]

```
template<typename _RandomAccessIter , typename _Integer , typename _UnaryPredicate >
constexpr _RandomAccessIter std::__search_n_aux (
 _RandomAccessIter __first,
 _RandomAccessIter __last,
 _Integer __count,
 _UnaryPredicate __unary_pred,
 std::random_access_iterator_tag)
```

This is an helper function for `search_n` overloaded for random access iterators.

Definition at line 227 of file `stl_algo.h`.



#### 3.10.4.29 \_\_stable\_partition\_adaptive()

```
template<typename _ForwardIterator , typename _Pointer , typename _Predicate , typename _Distance
>
_FForwardIterator std::__stable_partition_adaptive (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Predicate __pred,
 _Distance __len,
 _Pointer __buffer,
 _Distance __buffer_size)
```

This is a helper function... Requires `__first != __last` and `!__pred(__first)` and `__len == distance(__first, __last)`.

`!__pred(__first)` allows us to guarantee that we don't move-assign an element onto itself.

Definition at line 1548 of file `stl_algo.h`.

#### 3.10.4.30 \_\_unguarded\_insertion\_sort()

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::__unguarded_insertion_sort (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Compare __comp) [inline]
```

This is a helper function for the sort routine.

Definition at line 1868 of file `stl_algo.h`.

References `__unguarded_linear_insert()`.

Referenced by `__final_insertion_sort()`.

#### 3.10.4.31 \_\_unguarded\_linear\_insert()

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr void std::__unguarded_linear_insert (
 _RandomAccessIterator __last,
 _Compare __comp)
```

This is a helper function for the sort routine.

Definition at line 1824 of file `stl_algo.h`.

Referenced by `__unguarded_insertion_sort()`.

### 3.10.4.32 `__unguarded_partition()`

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr _RandomAccessIterator std::__unguarded_partition (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _RandomAccessIterator __pivot,
 _Compare __comp)
```

This is a helper function...

Definition at line 1903 of file `stl_algo.h`.

References `iter_swap()`.

Referenced by `__unguarded_partition_pivot()`.

### 3.10.4.33 `__unguarded_partition_pivot()`

```
template<typename _RandomAccessIterator , typename _Compare >
constexpr _RandomAccessIterator std::__unguarded_partition_pivot (
 _RandomAccessIterator __first,
 _RandomAccessIterator __last,
 _Compare __comp) [inline]
```

This is a helper function...

Definition at line 1925 of file `stl_algo.h`.

References `__move_median_to_first()`, and `__unguarded_partition()`.

Referenced by `__introsort_loop()`.

### 3.10.4.34 `__unique_copy()` [1/3]

```
template<typename _ForwardIterator , typename _OutputIterator , typename _BinaryPredicate >
constexpr _OutputIterator std::__unique_copy (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _OutputIterator __result,
 _BinaryPredicate __binary_pred,
 forward_iterator_tag ,
 output_iterator_tag)
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for forward iterators and output iterator as result.

Definition at line 1034 of file `stl_algo.h`.

**3.10.4.35 \_\_unique\_copy()** [2/3]

```
template<typename _InputIterator , typename _OutputIterator , typename _BinaryPredicate >
constexpr _OutputIterator std::__unique_copy (
 _InputIterator __first,
 _InputIterator __last,
 _OutputIterator __result,
 _BinaryPredicate __binary_pred,
 input_iterator_tag ,
 output_iterator_tag)
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and output iterator as result.

Definition at line 1064 of file `stl_algo.h`.

**3.10.4.36 \_\_unique\_copy()** [3/3]

```
template<typename _InputIterator , typename _ForwardIterator , typename _BinaryPredicate >
constexpr _ForwardIterator std::__unique_copy (
 _InputIterator __first,
 _InputIterator __last,
 _ForwardIterator __result,
 _BinaryPredicate __binary_pred,
 input_iterator_tag ,
 forward_iterator_tag)
```

This is an uglified `unique_copy(_InputIterator, _InputIterator, _OutputIterator, _BinaryPredicate)` overloaded for input iterators and forward iterator as result.

Definition at line 1097 of file `stl_algo.h`.

**3.10.4.37 \_Construct()**

```
template<typename _Tp , typename... _Args>
void std::_Construct (
 _Tp * __p,
 _Args &&... __args) [inline]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.

Definition at line 108 of file `stl_construct.h`.

**3.10.4.38** `_Destroy()` [1/3]

```
template<typename _ForwardIterator >
constexpr void std::_Destroy (
 _ForwardIterator __first,
 _ForwardIterator __last) [inline]
```

Destroy a range of objects. If the value\_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 171 of file `stl_construct.h`.

**3.10.4.39** `_Destroy()` [2/3]

```
template<typename _Tp >
constexpr void std::_Destroy (
 _Tp * __pointer) [inline]
```

Destroy the object pointed to by a pointer type.

Definition at line 135 of file `stl_construct.h`.

**3.10.4.40** `_Destroy()` [3/3]

```
template<typename _ForwardIterator , typename _Allocator >
void std::_Destroy (
 _ForwardIterator __first,
 _ForwardIterator __last,
 _Allocator & __alloc)
```

Destroy a range of objects using the supplied allocator. For non-default allocators we do not optimize away invocation of `destroy()` even if `_Tp` has a trivial destructor.

Definition at line 721 of file `bits/alloc_traits.h`.

References `__addressof()`, and `std::allocator_traits<_Alloc>::destroy()`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::~~vector()`.

**3.10.4.41** `_Destroy_n()`

```
template<typename _ForwardIterator , typename _Size >
constexpr _ForwardIterator std::_Destroy_n (
 _ForwardIterator __first,
 _Size __count) [inline]
```

Destroy a range of objects. If the value\_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.

Definition at line 220 of file `stl_construct.h`.

## 3.10.4.42 acos()

```
template<typename _Tp >
std::complex< _Tp > std::acos (
 const std::complex< _Tp > & __z) [inline]
```

acos(\_\_z) [8.1.2].

Definition at line 1634 of file complex.

## 3.10.4.43 acosh()

```
template<typename _Tp >
std::complex< _Tp > std::acosh (
 const std::complex< _Tp > & __z) [inline]
```

acosh(\_\_z) [8.1.5].

Definition at line 1750 of file complex.

## 3.10.4.44 advance()

```
template<typename _InputIterator , typename _Distance >
constexpr void std::advance (
 _InputIterator & __i,
 _Distance __n) [inline]
```

A generalization of pointer arithmetic.

## Parameters

|                     |                                          |
|---------------------|------------------------------------------|
| $\leftarrow$<br>__i | An input iterator.                       |
| $\leftarrow$<br>__n | The <i>delta</i> by which to change __i. |

## Returns

Nothing.

This increments *i* by *n*. For bidirectional and random access iterators, \_\_n may be negative, in which case \_\_i is decremented.

For random access iterators, this uses their + and – operations and are constant time. For other iterator classes they are linear time.

Definition at line 202 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__merge_without_buffer()`.

#### 3.10.4.45 `arg()`

```
template<typename _Tp >
__gnu_cxx::__promote<_Tp>::__type std::arg (
 _Tp __x) [inline]
```

Additional overloads [8.1.9].

Definition at line 1848 of file `complex`.

#### 3.10.4.46 `asin()`

```
template<typename _Tp >
std::complex<_Tp > std::asin (
 const std::complex<_Tp > & __z) [inline]
```

`asin(__z)` [8.1.3].

Definition at line 1670 of file `complex`.

#### 3.10.4.47 `asinh()`

```
template<typename _Tp >
std::complex<_Tp > std::asinh (
 const std::complex<_Tp > & __z) [inline]
```

`asinh(__z)` [8.1.6].

Definition at line 1789 of file `complex`.

#### 3.10.4.48 `atan()`

```
template<typename _Tp >
std::complex<_Tp > std::atan (
 const std::complex<_Tp > & __z) [inline]
```

`atan(__z)` [8.1.4].

Definition at line 1714 of file `complex`.

## 3.10.4.49 atanh()

```
template<typename _Tp >
std::complex< _Tp > std::atanh (
 const std::complex< _Tp > & __z) [inline]
```

atanh(\_\_z) [8.1.7].

Definition at line 1833 of file complex.

## 3.10.4.50 begin() [1/3]

```
template<typename _Container >
constexpr auto std::begin (
 _Container & __cont) -> decltype(__cont.begin()) [inline]
```

Return an iterator pointing to the first element of the container.

## Parameters

|        |            |
|--------|------------|
| __cont | Container. |
|--------|------------|

Definition at line 51 of file range\_access.h.

## 3.10.4.51 begin() [2/3]

```
template<typename _Container >
constexpr auto std::begin (
 const _Container & __cont) -> decltype(__cont.begin()) [inline]
```

Return an iterator pointing to the first element of the const container.

## Parameters

|        |            |
|--------|------------|
| __cont | Container. |
|--------|------------|

Definition at line 61 of file range\_access.h.

## 3.10.4.52 begin() [3/3]

```
template<typename _Tp , size_t _Nm>
constexpr _Tp* std::begin (
 _Tp(&) __arr[_Nm]) [inline], [noexcept]
```

Return an iterator pointing to the first element of the array.



## Parameters

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Definition at line 90 of file `range_access.h`.

## 3.10.4.53 boolalpha()

```
ios_base& std::boolalpha (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::boolalpha)`.

Definition at line 908 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::boolalpha`.

## 3.10.4.54 cbegin()

```
template<typename _Container >
constexpr auto std::cbegin (
 const _Container & __cont) -> decltype(std::begin(__cont)) [inline], [noexcept]
```

Return an iterator pointing to the first element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 119 of file `range_access.h`.

References `begin()`.

## 3.10.4.55 cend()

```
template<typename _Container >
constexpr auto std::cend (
 const _Container & __cont) -> decltype(std::end(__cont)) [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 130 of file `range_access.h`.

References `end()`.

**3.10.4.56 `const_pointer_cast()`**

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::const_pointer_cast (
 const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
```

**`const_pointer_cast`**

Definition at line 1588 of file `shared_ptr_base.h`.

**3.10.4.57 `crbegin()`**

```
template<typename _Container >
constexpr auto std::crbegin (
 const _Container & __cont) -> decltype(std::rbegin(__cont)) [inline]
```

Return a reverse iterator pointing to the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 221 of file `range_access.h`.

References `rbegin()`.

**3.10.4.58 `crend()`**

```
template<typename _Container >
constexpr auto std::crend (
 const _Container & __cont) -> decltype(std::rend(__cont)) [inline]
```

Return a reverse iterator pointing one past the first element of the const container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 231 of file `range_access.h`.

References `rend()`.

3.10.4.59 `dec()`

```
ios_base& std::dec (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::dec, ios_base::basefield)`.

Definition at line 1046 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, and `std::ios_base::dec`.

Referenced by operator `>>()`.

3.10.4.60 `defaultfloat()`

```
ios_base& std::defaultfloat (
 ios_base & __base) [inline]
```

Calls `base.unsetf(ios_base::floatfield)`

Definition at line 1099 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::floatfield`.

3.10.4.61 `distance()`

```
template<typename _InputIterator >
constexpr iterator_traits<_InputIterator>::difference_type std::distance (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A generalization of pointer arithmetic.

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

**Returns**

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.

For random access iterators, this uses their `+` and `-` operations and are constant time. For other iterator classes they are linear time.

Definition at line 138 of file `stl_iterator_base_funcs.h`.

References `__iterator_category()`.

Referenced by `__sample()`, `std::deque<_StateSeqT>::_M_range_initialize()`, `std::sub_match<_Bi_iter>::length()`, `__gnu_parallel::multiseq_partition()`, `__gnu_parallel::multiseq_selection()`, and `std::match_results<_Bi_iter>::position()`.

**3.10.4.62 `dynamic_pointer_cast()`**

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::dynamic_pointer_cast (
 const __shared_ptr<_Tp1, _Lp > & __r) [inline], [noexcept]
```

`dynamic_pointer_cast`

Definition at line 1601 of file `shared_ptr_base.h`.

**3.10.4.63 `end()` [1/3]**

```
template<typename _Container >
constexpr auto std::end (
 _Container & __cont) -> decltype(__cont.end()) [inline]
```

Return an iterator pointing to one past the last element of the container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 71 of file `range_access.h`.

**3.10.4.64 `end()` [2/3]**

```
template<typename _Container >
```

```
constexpr auto std::end (
 const _Container & __cont) -> decltype(__cont.end()) [inline]
```

Return an iterator pointing to one past the last element of the const container.

#### Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 81 of file `range_access.h`.

#### 3.10.4.65 `end()` [3/3]

```
template<typename _Tp , size_t _Nm>
constexpr _Tp* std::end (
 _Tp(&) __arr[_Nm]) [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the array.

#### Parameters

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Definition at line 100 of file `range_access.h`.

#### 3.10.4.66 `endl()`

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::endl (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↔streambuf.buffering> for more on this subject.

Definition at line 681 of file `ostream`.

#### 3.10.4.67 ends()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::ends (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Write a null character into the output sequence.

*Null character* is `CharT()` by definition. For `CharT` of `char`, this correctly writes the ASCII NUL character string terminator.

Definition at line 693 of file `ostream`.

#### 3.10.4.68 exchange()

```
template<typename _Tp , typename _Up = _Tp>
constexpr _Tp std::exchange (
 _Tp & __obj,
 _Up && __new_val) [inline]
```

Assign `__new_val` to `__obj` and return its previous value.

Definition at line 291 of file `utility`.

#### 3.10.4.69 fabs()

```
template<typename _Tp >
_Tp std::fabs (
 const std::complex< _Tp > & __z) [inline]
```

`fabs(__z)` [8.1.8].

Definition at line 1842 of file `complex`.

#### 3.10.4.70 fixed()

```
ios_base& std::fixed (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed, ios_base::floatfield)`.

Definition at line 1071 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::fixed`, and `std::ios_base::floatfield`.

#### 3.10.4.71 flush()

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::flush (
 basic_ostream< _CharT, _Traits > & __os) [inline]
```

Flushes the output stream.

This manipulator simply calls the stream's `flush()` member function.

Definition at line 703 of file ostream.

#### 3.10.4.72 from\_chars()

```
template<typename _Tp >
__detail::__integer_from_chars_result_type<_Tp> std::from_chars (
 const char * __first,
 const char * __last,
 _Tp & __value,
 int __base = 10)
```

`std::from_chars` for integral types.

Definition at line 595 of file charconv.

#### 3.10.4.73 get\_money()

```
template<typename _MoneyT >
_Get_money<_MoneyT> std::get_money (
 _MoneyT & __mon,
 bool __intl = false) [inline]
```

Extended manipulator for extracting money.

##### Parameters

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__mon</code>  | Either long double or a specialization of <code>basic_string</code> . |
| <code>__intl</code> | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator extracts `__mon`.

Definition at line 259 of file iomanip.

#### 3.10.4.74 `get_new_handler()`

```
new_handler std::get_new_handler () [noexcept]
```

Return the current new handler.

#### 3.10.4.75 `get_temporary_buffer()`

```
template<typename _Tp >
pair<_Tp*, ptrdiff_t> std::get_temporary_buffer (
 ptrdiff_t __len) [noexcept]
```

Allocates a temporary buffer.

##### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__len</code> | The number of objects of type <code>Tp</code> . |
|--------------------|-------------------------------------------------|

##### Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!

This function tries to obtain storage for `__len` adjacent `Tp` objects. The objects themselves are not constructed, of course. A `pair<>` is returned containing *the buffer's address and capacity (in the units of `sizeof(_Tp)`)*, or a pair of 0 values if no storage can be obtained. Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare `len` with the `.second` return value.

Provides the nothrow exception guarantee.

Definition at line 100 of file `stl_tempbuf.h`.

#### 3.10.4.76 `get_time()`

```
template<typename _CharT >
_Get_time<_CharT> std::get_time (
 std::tm * __tmb,
 const _CharT * __fmt) [inline]
```

Extended manipulator for extracting time.

This manipulator uses `time_get::get` to extract time. [ext.manip]



## Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__tmb</code> | struct to extract the time data to. |
| <code>__fmt</code> | format string.                      |

Definition at line 413 of file `iomanip`.

3.10.4.77 `getline()` [1/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 _CharT __delim)
```

Read a line from stream into a string.

## Parameters

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If `delim` was encountered, it is extracted but not stored into `__str`.

Definition at line 627 of file `vstring.tcc`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size()`.

3.10.4.78 `getline()` [2/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream<_CharT, _Traits>& std::getline (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Read a line from stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until '

' is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased. If end of line was encountered, it is extracted but not stored into `__str`.

Definition at line 2677 of file `vstring.h`.

References `getline()`, and `std::basic_ios<_CharT, _Traits >::widen()`.

**3.10.4.79 `getline()` [3/6]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream< _CharT, _Traits > & std::getline (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str,
 _CharT __delim)
```

Read a line from stream into a string.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__is</code>    | Input stream.                  |
| <code>__str</code>   | Buffer to store into.          |
| <code>__delim</code> | Character marking end of line. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until `__delim` is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If `__delim` is encountered, it is extracted but not stored into `__str`.

Definition at line 1540 of file `basic_string.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc >::max_size()`.

Referenced by `getline()`.

**3.10.4.80** `getline()` [4/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Read a line from stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `is` into `__str` until '  
' is found, the end of the stream is encountered, or `str.max_size()` is reached. Any previous contents of `__str` are erased. If end of line is encountered, it is extracted but not stored into `__str`.

Definition at line 6508 of file `basic_string.h`.

References `getline()`, and `std::basic_ios< _CharT, _Traits >::widen()`.

**3.10.4.81** `getline()` [5/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
 basic_istream< _CharT, _Traits > && __is,
 basic_string< _CharT, _Traits, _Alloc > & __str,
 _CharT __delim) [inline]
```

Read a line from an rvalue stream into a string.

Definition at line 6516 of file `basic_string.h`.

References `getline()`.

**3.10.4.82** `getline()` [6/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream<_CharT, _Traits>& std::getline (
 basic_istream< _CharT, _Traits > && __is,
 basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Read a line from an rvalue stream into a string.

Definition at line 6523 of file `basic_string.h`.

References `getline()`.

#### 3.10.4.83 hex()

```
ios_base& std::hex (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::hex, ios_base::basefield)`.

Definition at line 1054 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, and `std::ios_base::hex`.

#### 3.10.4.84 hexfloat()

```
ios_base& std::hexfloat (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::fixed|ios_basescientific, ios_base::floatfield)`

Definition at line 1091 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::fixed`, `std::ios_base::floatfield`, and `std::ios_base::scientific`.

#### 3.10.4.85 internal()

```
ios_base& std::internal (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::internal, ios_base::adjustfield)`.

Definition at line 1021 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::internal`.

#### 3.10.4.86 isalnum()

```
template<typename _CharT >
bool std::isalnum (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alnum, __c)`.

Definition at line 2623 of file `locale_facets.h`.

#### 3.10.4.87 isalpha()

```
template<typename _CharT >
bool std::isalpha (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::alpha, __c)`.

Definition at line 2599 of file `locale_facets.h`.

#### 3.10.4.88 isblank()

```
template<typename _CharT >
bool std::isblank (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::blank, __c)`.

Definition at line 2636 of file `locale_facets.h`.

#### 3.10.4.89 iscntrl()

```
template<typename _CharT >
bool std::iscntrl (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::cntrl, __c)`.

Definition at line 2581 of file `locale_facets.h`.

#### 3.10.4.90 isdigit()

```
template<typename _CharT >
bool std::isdigit (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::digit, __c)`.

Definition at line 2605 of file `locale_facets.h`.

#### 3.10.4.91 isgraph()

```
template<typename _CharT >
bool std::isgraph (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::graph, __c)`.

Definition at line 2629 of file `locale_facets.h`.

#### 3.10.4.92 islower()

```
template<typename _CharT >
bool std::islower (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::lower, __c)`.

Definition at line 2593 of file `locale_facets.h`.

#### 3.10.4.93 isprint()

```
template<typename _CharT >
bool std::isprint (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::print, __c)`.

Definition at line 2575 of file `locale_facets.h`.

#### 3.10.4.94 ispunct()

```
template<typename _CharT >
bool std::ispunct (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::punct, __c)`.

Definition at line 2611 of file `locale_facets.h`.

#### 3.10.4.95 isspace()

```
template<typename _CharT >
bool std::isspace (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::space, __c)`.

Definition at line 2569 of file `locale_facets.h`.

#### 3.10.4.96 isupper()

```
template<typename _CharT >
bool std::isupper (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::upper, __c)`.

Definition at line 2587 of file `locale_facets.h`.

#### 3.10.4.97 isxdigit()

```
template<typename _CharT >
bool std::isxdigit (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype.is(ctype_base::xdigit, __c)`.

Definition at line 2617 of file `locale_facets.h`.

#### 3.10.4.98 left()

```
ios_base& std::left (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::left, ios_base::adjustfield)`.

Definition at line 1029 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::left`.

#### 3.10.4.99 noboolalpha()

```
ios_base& std::noboolalpha (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::boolalpha).

Definition at line 916 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::boolalpha`.

#### 3.10.4.100 noshowbase()

```
ios_base& std::noshowbase (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::showbase).

Definition at line 932 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::showbase`.

#### 3.10.4.101 noshowpoint()

```
ios_base& std::noshowpoint (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::showpoint).

Definition at line 948 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::showpoint`.

#### 3.10.4.102 noshowpos()

```
ios_base& std::noshowpos (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::showpos).

Definition at line 964 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::showpos`.



#### 3.10.4.103 noskipws()

```
ios_base& std::noskipws (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::skipws).

Definition at line 980 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::skipws`.

#### 3.10.4.104 nunitbuf()

```
ios_base& std::nunitbuf (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::unitbuf).

Definition at line 1012 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::unitbuf`.

#### 3.10.4.105 nouppercase()

```
ios_base& std::nouppercase (
 ios_base & __base) [inline]
```

Calls base.unsetf(ios\_base::uppercase).

Definition at line 996 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::uppercase`.

#### 3.10.4.106 oct()

```
ios_base& std::oct (
 ios_base & __base) [inline]
```

Calls base.setf(ios\_base::oct, ios\_base::basefield).

Definition at line 1062 of file ios\_base.h.

References `__gnu_debug::__base()`, `std::ios_base::basefield`, and `std::ios_base::oct`.

**3.10.4.107 operator!=(())** [1/15]

```
template<typename _Tp , typename _Seq >
bool std::operator!=(
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Based on operator==.

Definition at line 343 of file stl\_stack.h.

**3.10.4.108 operator!=(())** [2/15]

```
template<typename _Tp , typename _Seq >
bool std::operator!=(
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Based on operator==.

Definition at line 368 of file stl\_queue.h.

**3.10.4.109 operator!=(())** [3/15]

```
template<typename _Res , typename... _Args>
bool std::operator!=(
 const function< _Res(_Args...)> & __f,
 nullptr_t) [inline], [noexcept]
```

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

**Returns**

`false` if the wrapper has no target, `true` otherwise

This function will not throw an exception.

Definition at line 699 of file std\_function.h.

**3.10.4.110 operator!=()** [4/15]

```
template<typename _Res , typename... _Args>
bool std::operator!= (
 nullptr_t ,
 const function< _Res(_Args...)> & __f) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 705 of file std\_function.h.

**3.10.4.111 operator!=()** [5/15]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator!= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x == y).

Definition at line 1016 of file stl\_multiset.h.

**3.10.4.112 operator!=()** [6/15]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator!= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x == y).

Definition at line 1030 of file stl\_set.h.

**3.10.4.113 operator!=()** [7/15]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator!= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator==.

Definition at line 1173 of file stl\_multimap.h.

**3.10.4.114 operator!=()** [8/15]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on operator==.

Definition at line 1476 of file forward\_list.h.

**3.10.4.115 operator!=()** [9/15]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator!= (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator==.

Definition at line 1508 of file stl\_map.h.

**3.10.4.116 operator!=()** [10/15]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Based on operator==.

Definition at line 1937 of file stl\_vector.h.

**3.10.4.117 operator!=()** [11/15]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

Based on operator==.

Definition at line 2057 of file stl\_list.h.

**3.10.4.118 operator!=()** [12/15]

```
template<typename _Tp , typename _Alloc >
bool std::operator!= (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline]
```

Based on operator==.

Definition at line 2286 of file `stl_deque.h`.

**3.10.4.119 operator!=()** [13/15]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Test difference of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 6239 of file `basic_string.h`.

**3.10.4.120 operator!=()** [14/15]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test difference of C string and string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__rhs.compare(__lhs) != 0`. False otherwise.

Definition at line 6252 of file `basic_string.h`.

**3.10.4.121 operator!=()** [15/15]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator!= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test difference of string and C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs.compare(__rhs) != 0`. False otherwise.

Definition at line 6264 of file `basic_string.h`.

**3.10.4.122 operator&()**

```
template<size_t _Nb>
bitset<_Nb> std::operator& (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [noexcept]
```

Global bitwise operations on bitsets.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A bitset.                                       |
| <code>__y</code> | A bitset of the same size as <code>__x</code> . |

**Returns**

A new bitset.

These should be self-explanatory.

Definition at line 1435 of file bitset.

**3.10.4.123 operator+()** [1/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs)
```

Concatenate two strings.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 6032 of file basic\_string.h.

**3.10.4.124 operator+()** [2/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs)
```

Concatenate C string and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with value of `__lhs` followed by `__rhs`.

Definition at line 1152 of file `basic_string.tcc`.

**3.10.4.125 operator+()** [3/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
 _CharT __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs)
```

Concatenate character and string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 1172 of file `basic_string.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`, and `std::basic_string< _CharT, _Traits, _Alloc >::size()`.

**3.10.4.126 operator+()** [4/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Concatenate string and C string.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |



**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 6069 of file `basic_string.h`.

**3.10.4.127 operator+()** [5/5]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string<_CharT, _Traits, _Alloc> std::operator+ (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 _CharT __rhs) [inline]
```

Concatenate string and character.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__lhs</code> | First string. |
| <code>__rhs</code> | Last string.  |

**Returns**

New string with `__lhs` followed by `__rhs`.

Definition at line 6085 of file `basic_string.h`.

**3.10.4.128 operator<()** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator< (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Stack ordering relation.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__x</code> | A stack.                                     |
| <code>__y</code> | A stack of the same type as <code>x</code> . |

**Returns**

True iff  $x$  is lexicographically less than  $\_\_y$ .

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with  $<$ , and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 337 of file `stl_stack.h`.

**3.10.4.129 operator<()** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator< (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Queue ordering relation.

**Parameters**

|         |                                   |
|---------|-----------------------------------|
| $\_\_x$ | A queue.                          |
| $\_\_y$ | A queue of the same type as $x$ . |

**Returns**

True iff  $\_\_x$  is lexicographically less than  $\_\_y$ .

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with  $<$ , and `std::lexicographical_compare()` is usually used to make the determination.

Definition at line 362 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

**3.10.4.130 operator<()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator< (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset ordering relation.

**Parameters**

|       |                                        |
|-------|----------------------------------------|
| $\_x$ | A multiset.                            |
| $\_y$ | A multiset of the same type as $\_x$ . |

**Returns**

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with  $<$ .

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1009 of file `std_multiset.h`.

**3.10.4.131 operator<()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator< (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set ordering relation.

**Parameters**

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A set.                            |
| $\_y$ | A set of the same type as $\_x$ . |

**Returns**

True iff  $\_x$  is lexicographically less than  $\_y$ .

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with  $<$ .

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1023 of file `std_set.h`.

**3.10.4.132 operator<()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator< (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap ordering relation.

**Parameters**

|                          |                                     |
|--------------------------|-------------------------------------|
| $\leftrightarrow$<br>__x | A multimap.                         |
| $\leftrightarrow$<br>__y | A multimap of the same type as __x. |

**Returns**

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <.

See std::lexicographical\_compare() for how the determination is made.

Definition at line 1166 of file stl\_multimap.h.

**3.10.4.133 operator<()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Forward list ordering relation.

**Parameters**

|                           |                                          |
|---------------------------|------------------------------------------|
| $\leftrightarrow$<br>__lx | A forward_list.                          |
| $\leftrightarrow$<br>__ly | A forward_list of the same type as __lx. |

**Returns**

True iff \_\_lx is lexicographically less than \_\_ly.

This is a total ordering relation. It is linear in the number of elements of the forward lists. The elements must be comparable with <.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1468 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::cbegin()`, `std::forward_list<_Tp, _Alloc>::cend()`, and `lexicographical_compare()`.

#### 3.10.4.134 `operator<()` [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator< (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map ordering relation.

##### Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__x</code> | A map.                                     |
| <code>__y</code> | A map of the same type as <code>x</code> . |

##### Returns

True iff `x` is lexicographically less than `y`.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1501 of file `stl_map.h`.

#### 3.10.4.135 `operator<()` [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Vector ordering relation.

##### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A vector.                                       |
| <code>__y</code> | A vector of the same type as <code>__x</code> . |

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the vectors. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 1930 of file `stl_vector.h`.

References `std::vector<_Tp, _Alloc>::begin()`, `std::vector<_Tp, _Alloc>::end()`, and `lexicographical_compare()`.

**3.10.4.136 operator<()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

List ordering relation.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__x</code> | A list.                                       |
| <code>__y</code> | A list of the same type as <code>__x</code> . |

**Returns**

True iff `__x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the lists. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 2050 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, `std::list<_Tp, _Alloc>::end()`, and `lexicographical_compare()`.

**3.10.4.137 operator<()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator< (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline]
```

Deque ordering relation.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A deque.                                       |
| <code>__y</code> | A deque of the same type as <code>__x</code> . |

**Returns**

True iff `x` is lexicographically less than `__y`.

This is a total ordering relation. It is linear in the size of the deques. The elements must be comparable with `<`.

See `std::lexicographical_compare()` for how the determination is made.

Definition at line 2279 of file `std_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, and `lexicographical_compare()`.

**3.10.4.138 operator<() [11/13]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Test if string precedes string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 6277 of file `basic_string.h`.

**3.10.4.139 operator<() [12/13]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string precedes C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 6290 of file `basic_string.h`.

**3.10.4.140 operator<>() [13/13]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator< (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test if C string precedes string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` precedes `__rhs`. False otherwise.

Definition at line 6302 of file `basic_string.h`.

**3.10.4.141 operator<<>() [1/14]**

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |



**Returns**`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 511 of file `ostream`.

**3.10.4.142 operator<<()** [2/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 517 of file `ostream`.

**3.10.4.143 operator<<()** [3/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 signed char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 523 of file `ostream`.

**3.10.4.144 operator<<()** [4/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 unsigned char __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 528 of file `ostream`.

**3.10.4.145 operator<<()** [5/14]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 const char * __s)
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

`out`

**Precondition**

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 321 of file `ostream.tcc`.

References `std::ios_base::badbit`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

**3.10.4.146 operator<<()** [6/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 const char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

\_\_s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 606 of file ostream.

**3.10.4.147 operator<<()** [7/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 const signed char * __s) [inline]
```

String inserters.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

**Returns**

out

**Precondition**

\_\_s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 619 of file ostream.

**3.10.4.148 operator<<()** [8/14]

```
template<typename _Traits >
basic_ostream<char, _Traits>& std::operator<< (
 basic_ostream< char, _Traits > & __out,
 const unsigned char * __s) [inline]
```

String inserters.

## Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

## Returns

`out`

## Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 624 of file `ostream`.

3.10.4.149 `operator<<()` [9/14]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream<_CharT, _Traits> & __out,
 const _CharT * __s) [inline]
```

String inserters.

## Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__out</code> | An output stream.   |
| <code>__s</code>   | A character string. |

## Returns

`out`

## Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

Definition at line 589 of file `ostream`.

**3.10.4.150 operator<<()** [10/14]

```
template<typename _CharT , typename _Traits >
basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream< _CharT, _Traits > & __out,
 _CharT __c) [inline]
```

Character inserters.

**Parameters**

|                    |                   |
|--------------------|-------------------|
| <code>__out</code> | An output stream. |
| <code>__c</code>   | A character.      |

**Returns**

`out`

Behaves like one of the formatted arithmetic inserters described in `std::basic_ostream`. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2]). `__out.width(0)` is then called.

If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

Definition at line 506 of file `ostream`.

**3.10.4.151 operator<<()** [11/14]

```
template<typename _Ostream , typename _Tp >
enable_if<__and<__not<is_lvalue_reference<_Ostream> >, __is_convertible_to_basic_ostream<_Ostream>, __is_insertable<__rvalue_ostream_type<_Ostream>, const _Tp&> >::value, __rvalue_ostream_type<_Ostream> >::type std::operator<< (
 _Ostream && __os,
 const _Tp & __x) [inline]
```

Generic inserter for rvalue stream.

**Parameters**

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__os</code> | An input stream.                          |
| <code>__x</code>  | A reference to the object being inserted. |

**Returns**

`os`

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

Definition at line 773 of file ostream.

#### 3.10.4.152 operator<<() [12/14]

```
template<class _CharT , class _Traits , size_t _Nb>
std::basic_ostream<_CharT, _Traits>& std::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Definition at line 1540 of file bitset.

#### 3.10.4.153 operator<<() [13/14]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Write string to a stream.

##### Parameters

|                    |                      |
|--------------------|----------------------|
| <code>__os</code>  | Output stream.       |
| <code>__str</code> | String to write out. |

##### Returns

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 2631 of file vstring.h.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**3.10.4.154** `operator<<()` [14/14]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_ostream<_CharT, _Traits>& std::operator<< (
 basic_ostream< _CharT, _Traits > & __os,
 const basic_string< _CharT, _Traits, _Alloc > & __str) [inline]
```

Write string to a stream.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__os</code>  | Output stream.       |
| <code>__str</code> | String to write out. |

**Returns**

Reference to the output stream.

Output characters of `__str` into `os` following the same rules as for writing a C string.

Definition at line 6468 of file `basic_string.h`.

**3.10.4.155** `operator<=()` [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator<= (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Based on `operator<`.

Definition at line 355 of file `stl_stack.h`.

**3.10.4.156** `operator<=()` [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator<= (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Based on `operator<`.

Definition at line 380 of file `stl_queue.h`.



**3.10.4.157 operator<=()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator<= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(y < x)

Definition at line 1030 of file stl\_multiset.h.

**3.10.4.158 operator<=()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator<= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(y < x)

Definition at line 1044 of file stl\_set.h.

**3.10.4.159 operator<=()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator<= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1187 of file stl\_multimap.h.

**3.10.4.160 operator<=()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on operator<.

Definition at line 1497 of file forward\_list.h.

**3.10.4.161 operator<=()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator<= (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1522 of file stl\_map.h.

**3.10.4.162 operator<=()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1949 of file stl\_vector.h.

**3.10.4.163 operator<=()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2069 of file stl\_list.h.

**3.10.4.164 operator<=()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator<= (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2298 of file stl\_deque.h.

**3.10.4.165 operator<=()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Test if string doesn't follow string.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6353 of file `basic_string.h`.

3.10.4.166 `operator<=()` [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't follow C string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

## Returns

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6366 of file `basic_string.h`.

3.10.4.167 `operator<=()` [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator<= (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test if C string doesn't follow string.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` doesn't follow `__rhs`. False otherwise.

Definition at line 6378 of file `basic_string.h`.

**3.10.4.168 operator==(** [1/16]

```
template<typename _StateT >
bool std::operator==(
 const fpos< _StateT > & __lhs,
 const fpos< _StateT > & __rhs) [inline]
```

Test if equivalent to another position.

Definition at line 222 of file `postypes.h`.

**3.10.4.169 operator==(** [2/16]

```
template<typename _Tp , typename _Seq >
bool std::operator==(
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Stack equality comparison.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A stack.                                       |
| <code>__y</code> | A stack of the same type as <code>__x</code> . |

**Returns**

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

Definition at line 319 of file `stl_stack.h`.

**3.10.4.170 operator==()** [3/16]

```
template<typename _Tp , typename _Seq >
bool std::operator== (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Queue equality comparison.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A queue.                                       |
| <code>__y</code> | A queue of the same type as <code>__x</code> . |

**Returns**

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.

Definition at line 344 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

**3.10.4.171 operator==()** [4/16]

```
template<typename _Res , typename... _Args>
bool std::operator== (
 const function< _Res(_Args...) > & __f,
 nullptr_t) [inline], [noexcept]
```

Compares a polymorphic function object wrapper against 0 (the NULL pointer).

**Returns**

`true` if the wrapper has no target, `false` otherwise

This function will not throw an exception.

Definition at line 680 of file `std_function.h`.

**3.10.4.172 operator==(** [5/16]

```
template<typename _Res , typename... _Args>
bool std::operator==(
 nullptr_t ,
 const function< _Res(_Args...)> & __f) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 687 of file std\_function.h.

**3.10.4.173 operator==(** [6/16]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator==(
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Multiset equality comparison.

**Parameters**

|       |                                        |
|-------|----------------------------------------|
| $\_x$ | A multiset.                            |
| $\_y$ | A multiset of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 971 of file stl\_multiset.h.

**3.10.4.174 operator==(** [7/16]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator==(
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Set equality comparison.

**Parameters**

|       |                              |
|-------|------------------------------|
| $\_x$ | A set.                       |
| $\_y$ | A set of the same type as x. |

**Returns**

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 985 of file `stl_set.h`.

**3.10.4.175 operator==()** [8/16]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator== (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Multimap equality comparison.

**Parameters**

|       |                                     |
|-------|-------------------------------------|
| $\_x$ | A multimap.                         |
| $\_y$ | A multimap of the same type as __x. |

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1128 of file `stl_multimap.h`.

**3.10.4.176** `operator==()` [9/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly)
```

Forward list equality comparison.



**Parameters**

|       |                                            |
|-------|--------------------------------------------|
| $\_x$ | A forward_list                             |
| $\_y$ | A forward_list of the same type as $\_x$ . |

**Returns**

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.

Definition at line 393 of file forward\_list.tcc.

References `std::forward_list<_Tp, _Alloc >::cbegin()`, and `std::forward_list<_Tp, _Alloc >::cend()`.

**3.10.4.177 operator==(** [10/16]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator==(
 const map<_Key, _Tp, _Compare, _Alloc > & __x,
 const map<_Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Map equality comparison.

**Parameters**

|       |                                   |
|-------|-----------------------------------|
| $\_x$ | A map.                            |
| $\_y$ | A map of the same type as $\_x$ . |

**Returns**

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1463 of file stl\_map.h.

**3.10.4.178 operator==()** [11/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Vector equality comparison.

**Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| $\leftarrow$<br>__x | A vector.                         |
| $\leftarrow$<br>__y | A vector of the same type as __x. |

**Returns**

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1892 of file stl\_vector.h.

References std::vector< \_Tp, \_Alloc >::begin(), std::vector< \_Tp, \_Alloc >::end(), equal(), and std::vector< \_Tp, \_Alloc >::size().

**3.10.4.179 operator==()** [12/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

List equality comparison.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| $\leftarrow$<br>__x | A list.                         |
| $\leftarrow$<br>__y | A list of the same type as __x. |

**Returns**

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 1995 of file `stl_list.h`.

References `std::list<_Tp, _Alloc>::begin()`, `std::list<_Tp, _Alloc>::end()`, and `std::list<_Tp, _Alloc>::size()`.

#### 3.10.4.180 `operator==()` [13/16]

```
template<typename _Tp , typename _Alloc >
bool std::operator== (
 const deque<_Tp, _Alloc > &__x,
 const deque<_Tp, _Alloc > &__y) [inline]
```

Deque equality comparison.

##### Parameters

|                          |                                  |
|--------------------------|----------------------------------|
| $\leftrightarrow$<br>__x | A deque.                         |
| $\leftrightarrow$<br>__y | A deque of the same type as __x. |

##### Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

Definition at line 2241 of file `stl_deque.h`.

References `std::deque<_Tp, _Alloc>::begin()`, `std::deque<_Tp, _Alloc>::end()`, `equal()`, and `std::deque<_Tp,  $\leftrightarrow$  _Alloc>::size()`.

#### 3.10.4.181 `operator==()` [14/16]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator== (
 const basic_string<_CharT, _Traits, _Alloc > &__lhs,
 const basic_string<_CharT, _Traits, _Alloc > &__rhs) [inline], [noexcept]
```

Test equivalence of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 6163 of file `basic_string.h`.

**3.10.4.182 operator==(** [15/16]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator==(
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test equivalence of string and C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs.compare(__rhs) == 0`. False otherwise.

Definition at line 6185 of file `basic_string.h`.

**3.10.4.183 operator==(** [16/16]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator==(
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test equivalence of C string and string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__rhs.compare(__lhs) == 0`. False otherwise.

Definition at line 6226 of file `basic_string.h`.

**3.10.4.184 operator>()** [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator> (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Based on `operator<`.

Definition at line 349 of file `stl_stack.h`.

**3.10.4.185 operator>()** [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator> (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Based on `operator<`.

Definition at line 374 of file `stl_queue.h`.

**3.10.4.186 operator>()** [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator> (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns `y < x`.

Definition at line 1023 of file `stl_multiset.h`.

**3.10.4.187 operator>()** [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator> (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns  $y < x$ .

Definition at line 1037 of file `stl_set.h`.

**3.10.4.188 operator>()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator> (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on `operator<`.

Definition at line 1180 of file `stl_multimap.h`.

**3.10.4.189 operator>()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on `operator<`.

Definition at line 1483 of file `forward_list.h`.

**3.10.4.190 operator>()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator> (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on `operator<`.

Definition at line 1515 of file `stl_map.h`.

**3.10.4.191 operator>()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1943 of file stl\_vector.h.

**3.10.4.192 operator>()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2063 of file stl\_list.h.

**3.10.4.193 operator>()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator> (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2292 of file stl\_deque.h.

**3.10.4.194 operator>()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Test if string follows string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 6315 of file `basic_string.h`.

**3.10.4.195 operator>() [12/13]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string follows C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.

Definition at line 6328 of file `basic_string.h`.

**3.10.4.196 operator>() [13/13]**

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator> (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test if C string follows string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` follows `__rhs`. False otherwise.



Definition at line 6340 of file basic\_string.h.

#### 3.10.4.197 operator>=() [1/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>= (
 const stack< _Tp, _Seq > & __x,
 const stack< _Tp, _Seq > & __y) [inline]
```

Based on operator<.

Definition at line 361 of file stl\_stack.h.

#### 3.10.4.198 operator>=() [2/13]

```
template<typename _Tp , typename _Seq >
bool std::operator>= (
 const queue< _Tp, _Seq > & __x,
 const queue< _Tp, _Seq > & __y) [inline]
```

Based on operator<.

Definition at line 386 of file stl\_queue.h.

#### 3.10.4.199 operator>=() [3/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator>= (
 const multiset< _Key, _Compare, _Alloc > & __x,
 const multiset< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x < y)

Definition at line 1037 of file stl\_multiset.h.

#### 3.10.4.200 operator>=() [4/13]

```
template<typename _Key , typename _Compare , typename _Alloc >
bool std::operator>= (
 const set< _Key, _Compare, _Alloc > & __x,
 const set< _Key, _Compare, _Alloc > & __y) [inline]
```

Returns !(x < y)

Definition at line 1051 of file stl\_set.h.

**3.10.4.201 operator>=()** [5/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator>= (
 const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 const multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1194 of file `stl_multimap.h`.

**3.10.4.202 operator>=()** [6/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on operator<.

Definition at line 1490 of file `forward_list.h`.

**3.10.4.203 operator>=()** [7/13]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
bool std::operator>= (
 const map< _Key, _Tp, _Compare, _Alloc > & __x,
 const map< _Key, _Tp, _Compare, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1529 of file `stl_map.h`.

**3.10.4.204 operator>=()** [8/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
 const vector< _Tp, _Alloc > & __x,
 const vector< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 1955 of file `stl_vector.h`.

**3.10.4.205 operator>=()** [9/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
 const list< _Tp, _Alloc > & __x,
 const list< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2075 of file stl\_list.h.

**3.10.4.206 operator>=()** [10/13]

```
template<typename _Tp , typename _Alloc >
bool std::operator>= (
 const deque< _Tp, _Alloc > & __x,
 const deque< _Tp, _Alloc > & __y) [inline]
```

Based on operator<.

Definition at line 2304 of file stl\_deque.h.

**3.10.4.207 operator>=()** [11/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Test if string doesn't precede string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6391 of file basic\_string.h.

**3.10.4.208 operator>=()** [12/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
 const basic_string< _CharT, _Traits, _Alloc > & __lhs,
 const _CharT * __rhs) [inline]
```

Test if string doesn't precede C string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | String.   |
| <code>__rhs</code> | C string. |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6404 of file `basic_string.h`.

**3.10.4.209 operator>=()** [13/13]

```
template<typename _CharT , typename _Traits , typename _Alloc >
bool std::operator>= (
 const _CharT * __lhs,
 const basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline]
```

Test if C string doesn't precede string.

**Parameters**

|                    |           |
|--------------------|-----------|
| <code>__lhs</code> | C string. |
| <code>__rhs</code> | String.   |

**Returns**

True if `__lhs` doesn't precede `__rhs`. False otherwise.

Definition at line 6416 of file `basic_string.h`.

**3.10.4.210 operator>>()** [1/11]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::operator>> (
```

```
basic_istream< _CharT, _Traits > & __in,
_CharT & __c)
```

Character extractors.

**Parameters**

|                          |                        |
|--------------------------|------------------------|
| $\leftrightarrow$<br>_in | An input stream.       |
| $\leftrightarrow$<br>_c  | A character reference. |

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.

Definition at line 931 of file `istream.tcc`.

References `std::ios_base::goodbit`.

**3.10.4.211 operator>>() [2/11]**

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
 basic_istream< char, _Traits > & __in,
 unsigned char & __c) [inline]
```

Character extractors.

**Parameters**

|                          |                        |
|--------------------------|------------------------|
| $\leftrightarrow$<br>_in | An input stream.       |
| $\leftrightarrow$<br>_c  | A character reference. |

**Returns**

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.



Definition at line 756 of file `istream`.

## 3.10.4.212 operator&gt;&gt;() [3/11]

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
 basic_istream< char, _Traits > & __in,
 signed char & __c) [inline]
```

Character extractors.

## Parameters

|                                                                                           |                        |
|-------------------------------------------------------------------------------------------|------------------------|
| <br>__in | An input stream.       |
| <br>__c  | A character reference. |

## Returns

in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in `__c`. Otherwise, sets failbit in the input stream.



Definition at line 761 of file `istream`.

## 3.10.4.213 operator&gt;&gt;() [4/11]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __in,
 _CharT * __s)
```

Character string extractors.

## Parameters

|                                                                                             |                                 |
|---------------------------------------------------------------------------------------------|---------------------------------|
| <br>__in | An input stream.                |
| <br>__s  | A pointer to a character array. |

## Returns

\_\_in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry

object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- `[27.6.1.2.3]/6`

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 963 of file `istream.tcc`.

References `std::ios_base::goodbit`.

#### 3.10.4.214 `operator>>()` [5/11]

```
template<>
basic_istream<char>& std::operator>> (
 basic_istream< char > & __in,
 char * __s)
```

Character string extractors.

##### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__in</code> | An input stream.                |
| <code>__s</code>  | A pointer to a character array. |

##### Returns

`__in`



Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

#### 3.10.4.215 operator>>() [6/11]

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
 basic_istream< char, _Traits > & __in,
 unsigned char * __s) [inline]
```

Character string extractors.

##### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__in</code> | An input stream.                |
| <code>__s</code>  | A pointer to a character array. |

##### Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise
- `n` is *the number of elements of the largest array of \**
- *char\_type that can store a terminating eos.*
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- `n-1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 803 of file `istream`.

#### 3.10.4.216 `operator>>()` [7/11]

```
template<class _Traits >
basic_istream<char, _Traits>& std::operator>> (
 basic_istream< char, _Traits > & __in,
 signed char * __s) [inline]
```

Character string extractors.

##### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__in</code> | An input stream.                |
| <code>__s</code>  | A pointer to a character array. |

##### Returns

`__in`

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array starting at `__s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `width()` otherwise

- *n* is the number of elements of the largest array of \*
- *char\_type* that can store a terminating eos.
- [27.6.1.2.3]/6

Characters are extracted and stored until one of the following happens:

- *n*-1 characters are stored
- EOF is reached
- the next character is whitespace according to the current locale
- the next character is a null byte (i.e., `charT()` )

`width(0)` is then called for the input stream.

If no characters are extracted, sets failbit.

Definition at line 808 of file `istream`.

#### 3.10.4.217 `operator>>()` [8/11]

```
template<typename _Istream, typename _Tp >
enable_if<__and<__not<is_lvalue_reference<_Istream> >, __is_convertible_to_basic_istream<_Istream>, __is_extractable<__rvalue_istream_type<_Istream>, _Tp&&> >::value, __rvalue_istream_type<_Istream> >::type std::operator>> (
 _Istream && __is,
 _Tp && __x) [inline]
```

Generic extractor for rvalue stream.

##### Parameters

|                   |                                       |
|-------------------|---------------------------------------|
| <code>__is</code> | An input stream.                      |
| <code>__x</code>  | A reference to the extraction target. |

##### Returns

`is`

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

Definition at line 980 of file `istream`.

**3.10.4.218 operator>>()** [9/11]

```
template<class _CharT , class _Traits , size_t _Nb>
std::basic_istream<_CharT, _Traits>& std::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept 0 and 1 characters, and will only extract as many digits as the bitset will hold.

Definition at line 1472 of file bitset.

**3.10.4.219 operator>>()** [10/11]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __is,
 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)
```

Read stream into a string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

**Returns**

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 552 of file vstring.tcc.

**3.10.4.220 operator>>()** [11/11]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_istream< _CharT, _Traits > & std::operator>> (
 basic_istream< _CharT, _Traits > & __is,
 basic_string< _CharT, _Traits, _Alloc > & __str)
```

Read stream into a string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__is</code>  | Input stream.         |
| <code>__str</code> | Buffer to store into. |

## Returns

Reference to the input stream.

Stores characters from `__is` into `__str` until whitespace is found, the end of the stream is encountered, or `str.max_size()` is reached. If `is.width()` is non-zero, that is the limit on the number of characters stored into `__str`. Any previous contents of `__str` are erased.

Definition at line 1468 of file `basic_string.tcc`.

3.10.4.221 `operator^()`

```
template<size_t _Nb>
bitset<_Nb> std::operator^ (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [noexcept]
```

Global bitwise operations on bitsets.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A bitset.                                       |
| <code>__y</code> | A bitset of the same size as <code>__x</code> . |

## Returns

A new bitset.

These should be self-explanatory.

Definition at line 1453 of file `bitset`.

3.10.4.222 `operator" | ()`

```
template<size_t _Nb>
bitset<_Nb> std::operator| (
 const bitset< _Nb > & __x,
 const bitset< _Nb > & __y) [inline], [noexcept]
```

Global bitwise operations on bitsets.

**Parameters**

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A bitset.                                       |
| <code>__y</code> | A bitset of the same size as <code>__x</code> . |

**Returns**

A new bitset.

These should be self-explanatory.

Definition at line 1444 of file `bitset`.

**3.10.4.223 `put_money()`**

```
template<typename _MoneyT >
_Put_money<_MoneyT> std::put_money (
 const _MoneyT & __mon,
 bool __intl = false) [inline]
```

Extended manipulator for inserting money.

**Parameters**

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__mon</code>  | Either long double or a specialization of <code>basic_string</code> . |
| <code>__intl</code> | A bool indicating whether international format is to be used.         |

Sent to a stream object, this manipulator inserts `__mon`.

Definition at line 306 of file `iomanip`.

**3.10.4.224 `put_time()`**

```
template<typename _CharT >
_Put_time<_CharT> std::put_time (
 const std::tm * __tmb,
 const _CharT * __fmt) [inline]
```

Extended manipulator for formatting time.

This manipulator uses `time_put::put` to format time. [ext.manip]

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__tmb</code> | struct tm time data to format. |
| <code>__fmt</code> | format string.                 |

Definition at line 358 of file `iomaniip`.

**3.10.4.225 quoted()**

```
template<typename _CharT >
auto std::quoted (
 const _CharT * __string,
 _CharT __delim = _CharT('\"'),
 _CharT __escape = _CharT('\\')) [inline]
```

Manipulator for quoted strings.

## Parameters

|                       |                                                       |
|-----------------------|-------------------------------------------------------|
| <code>__string</code> | String to quote.                                      |
| <code>__delim</code>  | Character to quote string with.                       |
| <code>__escape</code> | Escape character to escape itself or quote character. |

Definition at line 461 of file `iomaniip`.

**3.10.4.226 rbegin()** [1/4]

```
template<typename _Container >
constexpr auto std::rbegin (
 _Container & __cont) -> decltype(__cont.rbegin()) [inline]
```

Return a reverse iterator pointing to the last element of the container.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 141 of file `range_access.h`.

Referenced by `crbegin()`.

**3.10.4.227** `rbegin()` [2/4]

```
template<typename _Container >
constexpr auto std::rbegin (
 const _Container & __cont) -> decltype(__cont.rbegin()) [inline]
```

Return a reverse iterator pointing to the last element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 151 of file `range_access.h`.

**3.10.4.228** `rbegin()` [3/4]

```
template<typename _Tp , size_t _Nm>
constexpr reverse_iterator<_Tp*> std::rbegin (
 _Tp(&) __arr[_Nm]) [inline], [noexcept]
```

Return a reverse iterator pointing to the last element of the array.

**Parameters**

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Definition at line 181 of file `range_access.h`.

**3.10.4.229** `rbegin()` [4/4]

```
template<typename _Tp >
constexpr reverse_iterator<const _Tp*> std::rbegin (
 initializer_list< _Tp > __il) [inline], [noexcept]
```

Return a reverse iterator pointing to the last element of the `initializer_list`.

**Parameters**

|                   |                                 |
|-------------------|---------------------------------|
| <code>__il</code> | <code>initializer_list</code> . |
|-------------------|---------------------------------|

Definition at line 201 of file `range_access.h`.



**3.10.4.230** `rend()` [1/4]

```
template<typename _Container >
constexpr auto std::rend (
 _Container & __cont) -> decltype(__cont.rend()) [inline]
```

Return a reverse iterator pointing one past the first element of the container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 161 of file `range_access.h`.

Referenced by `crend()`.

**3.10.4.231** `rend()` [2/4]

```
template<typename _Container >
constexpr auto std::rend (
 const _Container & __cont) -> decltype(__cont.rend()) [inline]
```

Return a reverse iterator pointing one past the first element of the const container.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__cont</code> | Container. |
|---------------------|------------|

Definition at line 171 of file `range_access.h`.

**3.10.4.232** `rend()` [3/4]

```
template<typename _Tp , size_t _Nm>
constexpr reverse_iterator<_Tp*> std::rend (
 _Tp(&) __arr[_Nm]) [inline], [noexcept]
```

Return a reverse iterator pointing one past the first element of the array.

**Parameters**

|                    |        |
|--------------------|--------|
| <code>__arr</code> | Array. |
|--------------------|--------|

Definition at line 191 of file `range_access.h`.

3.10.4.233 `rend()` [4/4]

```
template<typename _Tp >
constexpr reverse_iterator<const _Tp*> std::rend (
 initializer_list< _Tp > __il) [inline], [noexcept]
```

Return a reverse iterator pointing one past the first element of the `initializer_list`.

## Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <code>__il</code> | <code>initializer_list</code> . |
|-------------------|---------------------------------|

Definition at line 211 of file `range_access.h`.

3.10.4.234 `replace_copy()`

```
template<typename _InputIterator , typename _OutputIterator , typename _Tp >
constexpr _OutputIterator std::replace_copy (
 _InputIterator __first,
 _InputIterator __last,
 _OutputIterator __result,
 const _Tp & __old_value,
 const _Tp & __new_value) [inline]
```

Copy a sequence, replacing each element of one value with another value.

## Parameters

|                          |                           |
|--------------------------|---------------------------|
| <code>__first</code>     | An input iterator.        |
| <code>__last</code>      | An input iterator.        |
| <code>__result</code>    | An output iterator.       |
| <code>__old_value</code> | The value to be replaced. |
| <code>__new_value</code> | The replacement value.    |

## Returns

The end of the output sequence, `result+(last-first)`.

Copies each element in the input range [`__first`,`__last`) to the output range [`__result`,`__result+(__last-__first)`) replacing elements equal to `__old_value` with `__new_value`.

Definition at line 3164 of file `stl_algo.h`.

## 3.10.4.235 resetiosflags()

```
_Resetiosflags std::resetiosflags (
 ios_base::fmtflags __mask) [inline]
```

Manipulator for `setf`.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator resets the specified flags, via `stream.setf(0, __mask)`.

Definition at line 66 of file `iosmanip`.

## 3.10.4.236 return\_temporary\_buffer()

```
template<typename _Tp >
void std::return_temporary_buffer (
 _Tp * __p) [inline]
```

The companion to `get_temporary_buffer()`.

## Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__p</code> | A buffer previously allocated by <code>get_temporary_buffer</code> . |
|------------------|----------------------------------------------------------------------|

## Returns

None.

Frees the memory pointed to by `__p`.

Definition at line 127 of file `stl_tempbuf.h`.

## 3.10.4.237 right()

```
ios_base& std::right (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::right, ios_base::adjustfield)`.

Definition at line 1037 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::adjustfield`, and `std::ios_base::right`.

**3.10.4.238 scientific()**

```
ios_base& std::scientific (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::scientific, ios_base::floatfield)`.

Definition at line 1079 of file `ios_base.h`.

References `__gnu_debug::__base()`, `std::ios_base::floatfield`, and `std::ios_base::scientific`.

**3.10.4.239 set\_new\_handler()**

```
new_handler std::set_new_handler (
 new_handler) throw ()
```

Takes a replacement handler as the argument, returns the previous handler.

**3.10.4.240 setbase()**

```
_Setbase std::setbase (
 int __base) [inline]
```

Manipulator for `setf`.

**Parameters**

|                     |                 |
|---------------------|-----------------|
| <code>__base</code> | A numeric base. |
|---------------------|-----------------|

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when `base` is 8, 10, or 16, accordingly, and to 0 if `__base` is any other value.

Definition at line 127 of file `iomanip`.

**3.10.4.241 setfill()**

```
template<typename _CharT >
_Setfill<_CharT> std::setfill (
 _CharT __c) [inline]
```

Manipulator for `fill`.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The new fill character. |
|------------------|-------------------------|

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

Definition at line 165 of file `iomanip`.

**3.10.4.242 setiosflags()**

```
_Setiosflags std::setiosflags (
 ios_base::fmtflags __mask) [inline]
```

Manipulator for `setf`.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__mask</code> | A format flags mask. |
|---------------------|----------------------|

Sent to a stream object, this manipulator sets the format flags to `__mask`.

Definition at line 96 of file `iomanip`.

**3.10.4.243 setprecision()**

```
_Setprecision std::setprecision (
 int __n) [inline]
```

Manipulator for `precision`.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__n</code> | The new precision. |
|------------------|--------------------|

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

Definition at line 195 of file `iomanip`.

#### 3.10.4.244 setw()

```
_Setw std::setw (
 int __n) [inline]
```

Manipulator for `width`.

##### Parameters

|                  |                |
|------------------|----------------|
| <code>__n</code> | The new width. |
|------------------|----------------|

Sent to a stream object, this manipulator calls `width(__n)` for that object.

Definition at line 225 of file `iosmanip`.

#### 3.10.4.245 showbase()

```
ios_base& std::showbase (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showbase)`.

Definition at line 924 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::showbase`.

#### 3.10.4.246 showpoint()

```
ios_base& std::showpoint (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpoint)`.

Definition at line 940 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::showpoint`.

#### 3.10.4.247 showpos()

```
ios_base& std::showpos (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::showpos)`.

Definition at line 956 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::showpos`.

**3.10.4.248 skipws()**

```
ios_base& std::skipws (
 ios_base & __base) [inline]
```

Calls base.setf(ios\_base::skipws).

Definition at line 972 of file ios\_base.h.

References `__gnu_debug::__base()`, and `std::ios_base::skipws`.

Referenced by `std::__detail::operator>>()`, and `operator>>()`.

**3.10.4.249 static\_pointer\_cast()**

```
template<typename _Tp , typename _Tp1 , _Lock_policy _Lp>
__shared_ptr<_Tp, _Lp> std::static_pointer_cast (
 const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
```

`static_pointer_cast`

Definition at line 1575 of file `shared_ptr_base.h`.

**3.10.4.250 swap()** [1/18]

```
template<typename _Res , typename... _Args>
void std::swap (
 function< _Res(_Args...)> & __x,
 function< _Res(_Args...)> & __y) [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.

This function will not throw an exception.

Definition at line 720 of file `std_function.h`.

**3.10.4.251 swap()** [2/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
 basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
 basic_stringbuf< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for stringbufs.

Definition at line 849 of file `sstream`.

**3.10.4.252 swap()** [3/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
 basic_istreamstream< _CharT, _Traits, _Allocator > & __x,
 basic_istreamstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for istringstreams.

Definition at line 856 of file sstream.

**3.10.4.253 swap()** [4/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
 basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
 basic_ostringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for ostringstreams.

Definition at line 863 of file sstream.

**3.10.4.254 swap()** [5/18]

```
template<class _CharT , class _Traits , class _Allocator >
void std::swap (
 basic_stringstream< _CharT, _Traits, _Allocator > & __x,
 basic_stringstream< _CharT, _Traits, _Allocator > & __y) [inline]
```

Swap specialization for stringstreams.

Definition at line 870 of file sstream.

**3.10.4.255 swap()** [6/18]

```
template<typename _Key , typename _Compare , typename _Alloc >
void std::swap (
 multiset< _Key, _Compare, _Alloc > & __x,
 multiset< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See std::multiset::swap().

Definition at line 1045 of file stl\_multiset.h.



**3.10.4.256 swap()** [7/18]

```
template<typename _Key , typename _Compare , typename _Alloc >
void std::swap (
 set< _Key, _Compare, _Alloc > & __x,
 set< _Key, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::set::swap()`.

Definition at line 1059 of file `stl_set.h`.

**3.10.4.257 swap()** [8/18]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
void std::swap (
 multimap< _Key, _Tp, _Compare, _Alloc > & __x,
 multimap< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::multimap::swap()`.

Definition at line 1202 of file `stl_multimap.h`.

**3.10.4.258 swap()** [9/18]

```
template<class _CharT , class _Traits >
void std::swap (
 basic_filebuf< _CharT, _Traits > & __x,
 basic_filebuf< _CharT, _Traits > & __y) [inline]
```

Swap specialization for filebufs.

Definition at line 1264 of file `fstream`.

**3.10.4.259 swap()** [10/18]

```
template<class _CharT , class _Traits >
void std::swap (
 basic_ifstream< _CharT, _Traits > & __x,
 basic_ifstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for ifstreams.

Definition at line 1271 of file `fstream`.

**3.10.4.260 swap()** [11/18]

```
template<class _CharT , class _Traits >
void std::swap (
 basic_ofstream< _CharT, _Traits > & __x,
 basic_ofstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for ofstreams.

Definition at line 1278 of file fstream.

**3.10.4.261 swap()** [12/18]

```
template<class _CharT , class _Traits >
void std::swap (
 basic_fstream< _CharT, _Traits > & __x,
 basic_fstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for fstreams.

Definition at line 1285 of file fstream.

**3.10.4.262 swap()** [13/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
 forward_list< _Tp, _Alloc > & __lx,
 forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See `std::forward_list::swap()`.

Definition at line 1505 of file `forward_list.h`.

**3.10.4.263 swap()** [14/18]

```
template<typename _Key , typename _Tp , typename _Compare , typename _Alloc >
void std::swap (
 map< _Key, _Tp, _Compare, _Alloc > & __x,
 map< _Key, _Tp, _Compare, _Alloc > & __y) [inline], [noexcept]
```

See `std::map::swap()`.

Definition at line 1537 of file `stl_map.h`.

**3.10.4.264 swap()** [15/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
 vector< _Tp, _Alloc > & __x,
 vector< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::vector::swap()`.

Definition at line 1962 of file `stl_vector.h`.

**3.10.4.265 swap()** [16/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
 list< _Tp, _Alloc > & __x,
 list< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::list::swap()`.

Definition at line 2082 of file `stl_list.h`.

**3.10.4.266 swap()** [17/18]

```
template<typename _Tp , typename _Alloc >
void std::swap (
 deque< _Tp, _Alloc > & __x,
 deque< _Tp, _Alloc > & __y) [inline], [noexcept]
```

See `std::deque::swap()`.

Definition at line 2311 of file `stl_deque.h`.

**3.10.4.267 swap()** [18/18]

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::swap (
 basic_string< _CharT, _Traits, _Alloc > & __lhs,
 basic_string< _CharT, _Traits, _Alloc > & __rhs) [inline], [noexcept]
```

Swap contents of two strings.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__lhs</code> | First string.  |
| <code>__rhs</code> | Second string. |

Exchanges the contents of `__lhs` and `__rhs` in constant time.

Definition at line 6430 of file `basic_string.h`.

#### 3.10.4.268 `tolower()`

```
template<typename _CharT >
_CharT std::tolower (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype::tolower(__c)`.

Definition at line 2649 of file `locale_facets.h`.

#### 3.10.4.269 `toupper()`

```
template<typename _CharT >
_CharT std::toupper (
 _CharT __c,
 const locale & __loc) [inline]
```

Convenience interface to `ctype::toupper(__c)`.

Definition at line 2643 of file `locale_facets.h`.

#### 3.10.4.270 `unitbuf()`

```
ios_base& std::unitbuf (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::unitbuf)`.

Definition at line 1004 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::unitbuf`.

#### 3.10.4.271 `uppercase()`

```
ios_base& std::uppercase (
 ios_base & __base) [inline]
```

Calls `base.setf(ios_base::uppercase)`.

Definition at line 988 of file `ios_base.h`.

References `__gnu_debug::__base()`, and `std::ios_base::uppercase`.

#### 3.10.4.272 ws()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::ws (
 basic_istream< _CharT, _Traits > & __is)
```

Quick and easy way to eat whitespace.

This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.

The current locale is used to distinguish whitespace characters.

Example:

```
MyClass mc;
std::cin >> std::ws >> mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a `std::basic_istream::sentry` inside your definition of operator>>.

Definition at line 1024 of file `istream.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::getloc()`, `std::basic_ios< _CharT, _Traits >::rdbuf()`, and `std::basic_ios< _CharT, _Traits >::setstate()`.

### 3.10.5 Variable Documentation

#### 3.10.5.1 \_\_ioint

```
ios_base::Init std::__ioint [static]
```

Linked to standard error (buffered)

Definition at line 74 of file `iostream`.

#### 3.10.5.2 cerr

```
ostream std::cerr
```

Linked to standard output.

### 3.10.5.3 cin

```
istream std::cin
```

Linked to standard input.

### 3.10.5.4 clog

```
ostream std::clog
```

Linked to standard error (unbuffered)

### 3.10.5.5 cout

```
ostream std::cout
```

Linked to standard input.

### 3.10.5.6 is\_nothrow\_swappable\_v

```
template<typename _Tp >
constexpr bool std::is_nothrow_swappable_v [inline]
```

is\_nothrow\_swappable\_v

Definition at line 2744 of file type\_traits.

### 3.10.5.7 is\_nothrow\_swappable\_with\_v

```
template<typename _Tp , typename _Up >
constexpr bool std::is_nothrow_swappable_with_v [inline]
```

is\_nothrow\_swappable\_with\_v

Definition at line 2828 of file type\_traits.

#### 3.10.5.8 is\_swappable\_v

```
template<typename _Tp >
constexpr bool std::is_swappable_v [inline]
```

is\_swappable\_v

Definition at line 2739 of file type\_traits.

#### 3.10.5.9 is\_swappable\_with\_v

```
template<typename _Tp , typename _Up >
constexpr bool std::is_swappable_with_v [inline]
```

is\_swappable\_with\_v

Definition at line 2823 of file type\_traits.

#### 3.10.5.10 wcerr

```
wostream std::wcerr
```

Linked to standard output.

#### 3.10.5.11 wcin

```
wistream std::wcin
```

Linked to standard error (buffered)

#### 3.10.5.12 wlog

```
wostream std::wlog
```

Linked to standard error (unbuffered)

#### 3.10.5.13 wcout

```
wostream std::wcout
```

Linked to standard input.

### 3.11 std::\_\_debug Namespace Reference

#### Classes

- class [bitset](#)
- class [deque](#)
- class [forward\\_list](#)
- class [list](#)
- class [map](#)
- class [multimap](#)
- class [multiset](#)
- class [set](#)
- class [unordered\\_map](#)
- class [unordered\\_multimap](#)
- class [unordered\\_multiset](#)
- class [unordered\\_set](#)
- class [vector](#)

#### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator!= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _←  
_Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >  
bool operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator >  
&__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered\_set< _Value,  
_Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp,  
_Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool operator!= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >  
bool operator!= (const unordered\_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered\_map<  
_Key, _Tp, _Hash, _Pred, _Alloc > &__y)`



- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`

- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`bool operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, size_t _Nb>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<size_t _Nb>`  
`bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<typename _Tp, size_t _Nm>`  
`enable_if< !::array_traits< _Tp, _Nm >::is_swappable::value >::type swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr void swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.<- swap(__two)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */i>)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */i>)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */i>)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */i>)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */i>)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */i>)`
- `template<typename _Tp, typename _Alloc >`  
`void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`
- `template<typename _Tp, typename _Alloc >`  
`void swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */i>)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc >`  
`void swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

### 3.11.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The `__glibcxx_requires_xxx` macros are merely wrappers around the `__glibcxx_check_xxx` wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

### 3.11.2 Function Documentation

#### 3.11.2.1 operator<=()

```
template<typename _Tp, typename _Alloc >
bool std::__debug::operator<= (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on `operator<`.

Definition at line 877 of file `debug/forward_list`.

## 3.11.2.2 operator&gt;()

```
template<typename _Tp , typename _Alloc >
bool std::__debug::operator> (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on operator<.

Definition at line 863 of file debug/forward\_list.

## 3.11.2.3 operator&gt;=()

```
template<typename _Tp , typename _Alloc >
bool std::__debug::operator>= (
 const forward_list< _Tp, _Alloc > & __lx,
 const forward_list< _Tp, _Alloc > & __ly) [inline]
```

Based on operator<.

Definition at line 870 of file debug/forward\_list.

## 3.11.2.4 swap()

```
template<typename _Tp , typename _Alloc >
void std::__debug::swap (
 forward_list< _Tp, _Alloc > & __lx,
 forward_list< _Tp, _Alloc > & __ly) [inline], [noexcept]
```

See std::forward\_list::swap().

Definition at line 885 of file debug/forward\_list.

## 3.12 std::\_\_detail Namespace Reference

## Classes

- struct [\\_BracketMatcher](#)
- class [\\_Compiler](#)
- struct [\\_Default\\_ranged\\_hash](#)
- struct [\\_Equality](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, false >](#)
- struct [\\_Equality< \\_Key, \\_Value, \\_Alloc, \\_ExtractKey, \\_Equal, \\_H1, \\_H2, \\_Hash, \\_RehashPolicy, \\_Traits, true >](#)
- class [\\_Executor](#)
- struct [\\_Hash\\_code\\_base](#)
- struct [\\_Hash\\_code\\_base< \\_Key, \\_Value, \\_ExtractKey, \\_H1, \\_H2, \\_Default\\_ranged\\_hash, false >](#)

- struct `_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Default_ranged_hash`, `true` >
- struct `_Hash_code_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Hash`, `false` >
- struct `_Hash_node`
- struct `_Hash_node`< `_Value`, `false` >
- struct `_Hash_node`< `_Value`, `true` >
- struct `_Hash_node_base`
- struct `_Hash_node_value_base`
- struct `_Hashtable_alloc`
- struct `_Hashtable_base`
- struct `_Hashtable_ebo_helper`
- struct `_Hashtable_ebo_helper`< `_Nm`, `_Tp`, `false` >
- struct `_Hashtable_ebo_helper`< `_Nm`, `_Tp`, `true` >
- struct `_Hashtable_traits`
- struct `_Insert`
- struct `_Insert`< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `false` >
- struct `_Insert`< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `true` >
- struct `_Insert_base`
- struct `_List_node_base`
- struct `_List_node_header`
- struct `_Local_const_iterator`
- struct `_Local_iterator`
- struct `_Local_iterator_base`
- struct `_Local_iterator_base`< `_Key`, `_Value`, `_ExtractKey`, `_H1`, `_H2`, `_Hash`, `true` >
- struct `_Map_base`
- struct `_Map_base`< `_Key`, `_Pair`, `_Alloc`, `_Select1st`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `false` >
- struct `_Map_base`< `_Key`, `_Pair`, `_Alloc`, `_Select1st`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `true` >
- struct `_Mask_range_hashing`
- struct `_Mod_range_hashing`
- struct `_Node_const_iterator`
- struct `_Node_iterator`
- struct `_Node_iterator_base`
- struct `_Power2_rehash_policy`
- struct `_Prime_rehash_policy`
- struct `_Quoted_string`
- struct `_Rehash_base`
- struct `_Rehash_base`< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `false_type` >
- struct `_Rehash_base`< `_Key`, `_Value`, `_Alloc`, `_ExtractKey`, `_Equal`, `_H1`, `_H2`, `_Hash`, `_RehashPolicy`, `_Traits`, `true_type` >
- class `_Scanner`
- class `_StateSeq`

## Typedefs

- template<typename `_Iter` , typename `_TraitsT` >  
using `__disable_if_contiguous_iter` = `__enable_if_t`< ! `__is_contiguous_iter`< `_Iter` >::value, `std::shared_ptr`<  
const `_NFA`< `_TraitsT` > > >
- template<typename `_Iter` , typename `_TraitsT` >  
using `__enable_if_contiguous_iter` = `__enable_if_t`< `__is_contiguous_iter`< `_Iter` >::value, `std::shared_ptr`<  
const `_NFA`< `_TraitsT` > > >
- template<typename `_Policy` >  
using `__has_load_factor` = `typename` `_Policy`::`__has_load_factor`

- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >`  
`using __hash_code_for_local_iter = _Hash_code_storage< __Hash_code_base< _Key, _Value, _ExtractKey,`  
`_H1, _H2, _Hash, false > >`
- `template<typename _Tp >`  
`using __integer_from_chars_result_type = enable_if_t< __or< __is_signed_integer< _Tp >, __is_↔`  
`unsigned_integer< _Tp >, is_same< char, remove_cv_t< _Tp > >>::value, from_chars_result >`
- `template<typename _Tp >`  
`using __integer_to_chars_result_type = enable_if_t< __or< __is_signed_integer< _Tp >, __is_unsigned_↔`  
`integer< _Tp >, is_same< char, remove_cv_t< _Tp > >>::value, to_chars_result >`
- `template<typename _Tp >`  
`using __unsigned_least_t = typename __to_chars_unsigned_type< _Tp >::type`
- `template<typename _CharT >`  
`using __Matcher = std::function< bool(_CharT)>`
- `typedef long __StateIdT`

## Enumerations

- `enum __Opcode : int {`  
`__S_opcode_unknown, __S_opcode_alternative, __S_opcode_repeat, __S_opcode_backref,`  
`__S_opcode_line_begin_assertion, __S_opcode_line_end_assertion, __S_opcode_word_boundary, __S_↔`  
`opcode_subexpr_lookahead,`  
`__S_opcode_subexpr_begin, __S_opcode_subexpr_end, __S_opcode_dummy, __S_opcode_match,`  
`__S_opcode_accept }`
- `enum __RegexExecutorPolicy : int { __S_auto, __S_alternate }`

## Functions

- `template<typename _Up, typename _Tp >`  
`constexpr _Up __absu (_Tp __val)`
- `template<typename _Up >`  
`void __absu (bool)=delete`
- `std::size_t __clp2 (std::size_t __n) noexcept`
- `template<typename _TraitsT, typename _FwdIter >`  
`__enable_if_contiguous_iter< _FwdIter, _TraitsT > __compile_nfa (_FwdIter __first, _FwdIter __last, const type-↔`  
`name _TraitsT::locale_type & __loc, regex_constants::syntax_option_type __flags)`
- `template<typename _TraitsT, typename _FwdIter >`  
`__disable_if_contiguous_iter< _FwdIter, _TraitsT > __compile_nfa (_FwdIter __first, _FwdIter __last, const ↔`  
`typename _TraitsT::locale_type & __loc, regex_constants::syntax_option_type __flags)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type __distance_fw (_Iterator __first, _Iterator __last)`
- `template<typename _Container, typename _Predicate >`  
`_Container::size_type __erase_nodes_if (_Container & __cont, _Predicate __pred)`
- `template<typename _ValT, typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & __extract_params (basic_istream< _CharT, _Traits > & __is, vector< _ValT ↔`  
`> & __vals, size_t __n)`
- `template<typename _Tp >`  
`bool __from_chars_alnum (const char * & __first, const char * __last, _Tp & __val, int __base)`

- constexpr unsigned char **\_\_from\_chars\_alpha\_to\_num** (char \_\_c)
- template<typename \_Tp >  
bool **\_\_from\_chars\_binary** (const char \*\_\_first, const char \*\_\_last, \_Tp &\_\_val)
- template<typename \_Tp >  
bool **\_\_from\_chars\_digit** (const char \*\_\_first, const char \*\_\_last, \_Tp &\_\_val, int \_\_base)
- template<typename \_Tp >  
constexpr \_Tp **\_\_gcd** (\_Tp \_\_m, \_Tp \_\_n)
- template<typename \_Tp >  
constexpr \_Tp **\_\_lcm** (\_Tp \_\_m, \_Tp \_\_n)
- template<typename \_InputIterator, typename \_OutputIterator, typename \_Tp >  
\_OutputIterator **\_\_normalize** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_OutputIterator \_\_result, const \_Tp &\_\_factor)
- template<typename \_Tp >  
bool **\_\_raise\_and\_add** (\_Tp &\_\_val, int \_\_base, unsigned char \_\_c)
- template<typename \_Bilter, typename \_Alloc, typename \_CharT, typename \_TraitsT, \_RegexExecutorPolicy \_\_policy, bool \_\_match\_mode>  
bool **\_\_regex\_algo\_impl** (\_Bilter \_\_s, \_Bilter \_\_e, match\_results< \_Bilter, \_Alloc > &\_\_m, const basic\_regex< \_CharT, \_TraitsT > &\_\_re, regex\_constants::match\_flag\_type \_\_flags)
- template<typename \_Tp >  
void **\_\_return\_temporary\_buffer** (\_Tp \*\_\_p, size\_t \_\_len)
- template<typename \_Tp >  
**to\_chars\_result** **\_\_to\_chars** (char \*\_\_first, char \*\_\_last, \_Tp \_\_val, int \_\_base) noexcept
- template<typename \_Tp >  
\_\_integer\_to\_chars\_result\_type< \_Tp > **\_\_to\_chars\_10** (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
void **\_\_to\_chars\_10\_impl** (char \*\_\_first, unsigned \_\_len, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
\_\_integer\_to\_chars\_result\_type< \_Tp > **\_\_to\_chars\_16** (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
\_\_integer\_to\_chars\_result\_type< \_Tp > **\_\_to\_chars\_2** (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
\_\_integer\_to\_chars\_result\_type< \_Tp > **\_\_to\_chars\_8** (char \*\_\_first, char \*\_\_last, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
constexpr unsigned **\_\_to\_chars\_len** (\_Tp \_\_value, int \_\_base) noexcept
- template<typename \_Tp >  
constexpr unsigned **\_\_to\_chars\_len\_2** (\_Tp \_\_value) noexcept
- template<typename \_Tp >  
bool **\_\_Power\_of\_2** (\_Tp \_\_x)
- template<typename \_Value, bool \_Cache\_hash\_code>  
bool **operator!=** (const **\_\_Node\_iterator\_base**< \_Value, \_Cache\_hash\_code > &\_\_x, const **\_\_Node\_iterator\_base**< \_Value, \_Cache\_hash\_code > &\_\_y) noexcept
- template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_H1, typename \_H2, typename \_Hash, bool \_\_cache>  
bool **operator!=** (const **\_\_Local\_iterator\_base**< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_x, const **\_\_Local\_iterator\_base**< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache > &\_\_y)
- template<typename \_CharT, typename \_Traits >  
**std::basic\_ostream**< \_CharT, \_Traits > & **operator<<** (**std::basic\_ostream**< \_CharT, \_Traits > &\_\_os, const **\_\_Quoted\_string**< const \_CharT \*, \_CharT > &\_\_str)
- template<typename \_CharT, typename \_Traits, typename \_String >  
**std::basic\_ostream**< \_CharT, \_Traits > & **operator<<** (**std::basic\_ostream**< \_CharT, \_Traits > &\_\_os, const **\_\_Quoted\_string**< \_String, \_CharT > &\_\_str)
- template<typename \_Value, bool \_Cache\_hash\_code>  
bool **operator==** (const **\_\_Node\_iterator\_base**< \_Value, \_Cache\_hash\_code > &\_\_x, const **\_\_Node\_iterator\_base**< \_Value, \_Cache\_hash\_code > &\_\_y) noexcept



- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool operator==(const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__x,`  
`const _Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache> &__y)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`std::basic_istream<_CharT, _Traits> & operator>> (std::basic_istream<_CharT, _Traits> &__is, const`  
`_Quoted_string<basic_string<_CharT, _Traits, _Alloc> &, _CharT> &__str)`

## Variables

- `static const _StateIdT _S_invalid_state_id`

### 3.12.1 Detailed Description

Implementation details not part of the namespace std interface.

### 3.12.2 Function Documentation

#### 3.12.2.1 \_\_from\_chars\_alnum()

```
template<typename _Tp >
bool std::__detail::__from_chars_alnum (
 const char *& __first,
 const char * __last,
 _Tp & __val,
 int __base)
```

std::from\_chars implementation for integers in bases 11 to 26.

Definition at line 560 of file charconv.

#### 3.12.2.2 \_\_from\_chars\_binary()

```
template<typename _Tp >
bool std::__detail::__from_chars_binary (
 const char *& __first,
 const char * __last,
 _Tp & __val)
```

std::from\_chars implementation for integers in base 2.

Definition at line 414 of file charconv.

### 3.12.2.3 `__from_chars_digit()`

```
template<typename _Tp >
bool std::__detail::__from_chars_digit (
 const char *& __first,
 const char * __last,
 _Tp & __val,
 int __base)
```

`std::from_chars` implementation for integers in bases 3 to 10.

Definition at line 441 of file `charconv`.

### 3.12.2.4 `operator<<()` [1/2]

```
template<typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& std::__detail::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const _Quoted_string< const _CharT *, _CharT > & __str)
```

Insertion for quoted strings.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 2344 `quoted()`'s interaction with padding is unclear

Definition at line 93 of file `quoted_string.h`.

References `std::basic_ostringstream< _CharT, _Traits, _Alloc >::str()`.

### 3.12.2.5 `operator<<()` [2/2]

```
template<typename _CharT , typename _Traits , typename _String >
std::basic_ostream<_CharT, _Traits>& std::__detail::operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const _Quoted_string< _String, _CharT > & __str)
```

Insertion for quoted strings.

`_GLIBCXX_RESOLVE_LIB_DEFECTS` DR 2344 `quoted()`'s interaction with padding is unclear

Definition at line 117 of file `quoted_string.h`.

References `std::basic_ostringstream< _CharT, _Traits, _Alloc >::str()`.

## 3.12.2.6 operator&gt;&gt;()

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_istream<_CharT, _Traits>& std::__detail::operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str)
```

Extractor for delimited strings. The left and right delimiters can be different.

Definition at line 139 of file quoted\_string.h.

References `std::basic_ios< _CharT, _Traits >::clear()`, `std::ios_base::flags()`, `std::basic_ios< _CharT, _Traits >::good()`, `std::ios_base::setf()`, `std::skipws()`, and `std::basic_istream< _CharT, _Traits >::unget()`.

## 3.13 std::\_\_parallel Namespace Reference

## Classes

- struct [\\_CRandNumber](#)

## Functions

- template<typename \_Iter , typename \_Tp , typename \_Tag >  
\_Tp **accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_Tag)
- template<typename \_Iter , typename \_Tp , typename \_IteratorTag >  
\_Tp **accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_IteratorTag)
- template<typename \_Iter , typename \_Tp , typename \_BinaryOper , typename \_Tag >  
\_Tp **accumulate\_switch** (\_Iter, \_Iter, \_Tp, \_BinaryOper, \_Tag)
- template<typename \_Iter , typename \_Tp , typename \_BinaryOperation , typename \_IteratorTag >  
\_Tp **accumulate\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op, \_IteratorTag)
- template<typename \_RAIter , typename \_Tp , typename \_BinaryOper >  
\_Tp **accumulate\_switch** (\_RAIter, \_RAIter, \_Tp, \_BinaryOper, [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::Parallelism](#)  
[\\_\\_parallelism=\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_RAIter , typename \_Tp , typename \_BinaryOperation >  
\_Tp **accumulate\_switch** (\_RAIter \_\_begin, \_RAIter \_\_end, \_Tp \_\_init, \_BinaryOperation \_\_binary\_op,  
[random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism\_tag)
- template<typename \_Iter , typename \_OIter , typename \_BinaryOper , typename \_Tag1 , typename \_Tag2 >  
\_OIter **adjacent\_difference\_switch** (\_Iter, \_Iter, \_OIter, \_BinaryOper, \_Tag1, \_Tag2)
- template<typename \_Iter , typename \_OIter , typename \_BinaryOper >  
\_OIter **adjacent\_difference\_switch** (\_Iter, \_Iter, \_OIter, \_BinaryOper, [random\\_access\\_iterator\\_tag](#),  
[random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::Parallelism](#) \_\_parallelism=[\\_\\_gnu\\_parallel::parallel\\_unbalanced](#))
- template<typename \_Iter , typename \_OutputIterator , typename \_BinaryOperation , typename \_IteratorTag1 , typename \_IteratorTag2 >  
\_OutputIterator **adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_Binary←  
Operation \_\_bin\_op, \_IteratorTag1, \_IteratorTag2)
- template<typename \_Iter , typename \_OutputIterator , typename \_BinaryOperation >  
\_OutputIterator **adjacent\_difference\_switch** (\_Iter \_\_begin, \_Iter \_\_end, \_OutputIterator \_\_result, \_Binary←  
Operation \_\_bin\_op, [random\\_access\\_iterator\\_tag](#), [random\\_access\\_iterator\\_tag](#), [\\_\\_gnu\\_parallel::Parallelism](#) ←  
\_\_parallelism\_tag)
- template<typename \_Filter , typename \_IterTag >  
\_Filter **adjacent\_find\_switch** (\_Filter, \_Filter, \_IterTag)

- `template<typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_Filter __adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _RAIter >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, random\_access\_iterator\_tag)`
- `template<typename _FIterator, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _IteratorTag)`
- `template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator __adjacent_find_switch (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _BinaryPredicate >`  
`_RAIter __adjacent_find_switch (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`iterator\_traits< _RAIter >::difference_type __count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`iterator\_traits< _Iter >::difference_type __count_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`iterator\_traits< _RAIter >::difference_type __count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`iterator\_traits< _Iter >::difference_type __count_switch (_Iter __begin, _Iter __end, const _Tp & __value, _IteratorTag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool __equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool __equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >`  
`_RAIter __find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random\_access\_iterator\_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Iter __find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag >`  
`_RAIter __find_first_of_switch (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random\_access\_iterator\_tag, _IteratorTag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Iter __find_first_of_switch (_Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >`  
`_Iter __find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`

- `template<typename _Iter, typename _Predicate, typename _IteratorTag >`  
`_Iter find_if_switch (_Iter __begin, _Iter __end, _Predicate __pred, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter find_switch (_Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter find_switch (_RAIter __begin, _RAIter __end, const _Tp &__val, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >`  
`_Iter find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function for_each_switch (_Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >`  
`_Function for_each_switch (_Iter, _Iter, _Function, _IterTag)`
- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`  
`_OutputIterator generate_n_switch (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _Filterator, typename _Generator, typename _IteratorTag >`  
`void generate_switch (_Filterator __begin, _Filterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism=__gnu_parallel::parallel_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`  
`_Tp inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Tp inner_product_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag >`  
`_Filterator max_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`

- `template<typename _RAIter, typename _Compare >`  
`_RAIter __max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2,  
typename _IterTag3 >  
_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter __merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2,  
typename _IteratorTag3 >  
_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result,  
_Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator __merge_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result,  
_Compare __comp, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter __min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _Filterator, typename _Compare, typename _IteratorTag >`  
`_Filterator __min_element_switch (_Filterator __begin, _Filterator __end, _Compare __comp, _IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter __min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, random\_access\_iterator\_tag,  
gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred,  
_IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,  
_Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,  
_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > __mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2,  
_RAIter2 __end2, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > __mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter __partial_sum_switch (_Iter, _Iter, _OIter, _BinaryOper, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
_IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator __partial_sum_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`  
`_Filter __partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _IteratorTag >`  
`_Filterator __partition_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, _IteratorTag)`

- `template<typename _RAIter, typename _Predicate >`  
`_RAIter __partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void __replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void __replace_if_switch (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void __replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void __replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _Filterator, typename _Tp, typename _IteratorTag >`  
`void __replace_switch (_Filterator __begin, _Filterator __end, const _Tp & __old_value, const _Tp & __new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void __replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp & __new_value, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter __search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random\_access\_iterator\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter __search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter __search_n_switch (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, random\_access\_iterator\_tag)`
- `template<typename _Filterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_Filterator __search_n_switch (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp & __val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 __search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 __search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 __search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Filterator1, typename _Filterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_Filterator1 __search_switch (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`

- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter __set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2,`  
`typename _IterTag3 >`  
`_OIter __set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _Iter←`  
`Tag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_intersection_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2,`  
`_OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter __set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2,`  
`typename _IterTag3 >`  
`_OIter __set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _Iter←`  
`Tag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_symmetric_difference_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2`  
`__end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter __set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ←`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2,`  
`typename _IterTag3 >`  
`_OIter __set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _Iter←`  
`Tag2, _IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename`  
`_IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator __set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Iter←`  
`OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter __set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 ←`  
`__end2, _Output_RAlter __result, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2,`  
`typename _IterTag3 >`  
`_OIter __set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_OIter __transform1_switch (_Iter, _Iter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOIter, typename _UnaryOperation >`  
`_RAOIter __transform1_switch (_RAIter, _RAIter, _RAOIter, _UnaryOperation, random\_access\_iterator\_tag,  
random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_balanced)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`  
`_RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation`  
`__unary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __←`  
`parallelism_tag)`



- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 > _RAIter2 __transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation > _RAIter3 __transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism=gnu\_parallel::parallel\_bal)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3 > _OIter __transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation > _RAIter3 __transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random\_access\_iterator\_tag, random\_access\_iterator\_tag, random\_access\_iterator\_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 > _OutputIterator __transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 > _OutputIterator __unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate > _RandomAccessOutputIterator __unique_copy_switch (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 > _OIter __unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate > _RandomAccess_OIter __unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random\_access\_iterator\_tag, random\_access\_iterator\_tag)`
- `template<typename _Iter, typename _Tp > _Tp accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp > _Tp accumulate (_Iter, _Iter, _Tp, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp > _Tp accumulate (_Iter, _Iter, _Tp, gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation > _Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper > _Tp accumulate (_Iter, _Iter, _Tp, _BinaryOper, gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation > _Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation > _Tp accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OIter > _OIter adjacent_difference (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper > _OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OIter > _OIter adjacent_difference (_Iter, _Iter, _OIter, gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter adjacent_difference (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator\_traits< _Iter >::difference_type count (_Iter __begin, _Iter __end, const _Tp &__value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator\_traits< _Iter >::difference_type count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr bool equal ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter find_first_of ( _Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Filterator >`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate >`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filterator, typename _BinaryPredicate >`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _Filterator >`  
`_Iter find_first_of ( _Iter __begin1, _Iter __end1, _Filterator __begin2, _Filterator __end2)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function for_each ( _Iterator __begin, _Iterator __end, _Function __f)`

- `template<typename _Iter, typename _Function >`  
`_Function for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void generate (_FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator generate_n (_OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter)`

- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter max_element (_Filter, _Filter, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::\_Parallelism)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵  
result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵  
result, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵  
result, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator merge (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __↵  
result)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter min_element (_Filter, _Filter, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Filter, typename _Compare >`  
`_Filter min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __binary←  
op)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter partition (_Filter, _Filter, _Predicate)`
- `template<typename _Filterator, typename _Predicate >`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Predicate >`  
`_Filterator partition (_Filterator __begin, _Filterator __end, _Predicate __pred)`
- `template<typename _RAlter >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator, typename _Tp >`  
`void replace (_Filterator __begin, _Filterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`



- `template<typename _Filterator , typename _Predicate , typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filterator , typename _Predicate , typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filterator , typename _Predicate , typename _Tp >`  
`void replace_if (_Filterator __begin, _Filterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _Filter1 , typename _Filter2 >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1 , typename _Filter2 >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1 , typename _Filter2 , typename _BiPredicate >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter1 , typename _Filter2 , typename _BiPredicate >`  
`_Filter1 search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filterator1 , typename _Filterator2 >`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _Filterator1 , typename _Filterator2 >`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2)`
- `template<typename _Filterator1 , typename _Filterator2 , typename _BinaryPredicate >`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _↵`  
`BinaryPredicate __pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator1 , typename _Filterator2 , typename _BinaryPredicate >`  
`_Filterator1 search (_Filterator1 __begin1, _Filterator1 __end1, _Filterator2 __begin2, _Filterator2 __end2, _↵`  
`BinaryPredicate __pred)`
- `template<typename _Filter , typename _Integer , typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)`
- `template<typename _Filter , typename _Integer , typename _Tp , typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)`
- `template<typename _Filter , typename _Integer , typename _Tp >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter , typename _Integer , typename _Tp , typename _BiPredicate >`  
`_Filter search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Filterator , typename _Integer , typename _Tp >`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator , typename _Integer , typename _Tp , typename _BinaryPredicate >`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate`  
`__binary_pred, __gnu_parallel::sequential_tag)`
- `template<typename _Filterator , typename _Integer , typename _Tp >`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _Filterator , typename _Integer , typename _Tp , typename _BinaryPredicate >`  
`_Filterator search_n (_Filterator __begin, _Filterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator >`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, __gnu_parallel::sequential_tag)`
- `template<typename _Iter1 , typename _Iter2 , typename _OutputIterator , typename _Predicate >`  
`_OutputIterator set_difference (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Output↵`  
`Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)`



- [illegible]

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator set_union (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator`  
`__out, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`

- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result,  $\leftrightarrow$  BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result,  $\leftrightarrow$  BinaryOperation __binary_op, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result,  $\leftrightarrow$  BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`

- `template<typename _Iter, typename _OIter >  
_OIter unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >  
_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >  
_OIter unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _Predicate >  
_OIter unique_copy (_Iter, _Iter, _OIter, _Predicate)`

### 3.13.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

## 3.14 `std::chrono` Namespace Reference

### Classes

- struct [duration](#)
- struct [duration\\_values](#)
- struct [time\\_point](#)
- struct [treat\\_as\\_floating\\_point](#)

### Typedefs

- using [hours](#) = [duration](#)< int64\_t, [ratio](#)< 3600 > >
- using [microseconds](#) = [duration](#)< int64\_t, [micro](#) >
- using [milliseconds](#) = [duration](#)< int64\_t, [milli](#) >
- using [minutes](#) = [duration](#)< int64\_t, [ratio](#)< 60 > >
- using [nanoseconds](#) = [duration](#)< int64\_t, [nano](#) >
- using [seconds](#) = [duration](#)< int64\_t >

### Functions

- `template<typename _ToDur, typename _Rep, typename _Period >  
constexpr \_\_enable\_if\_is\_duration< _ToDur > duration\_cast (const duration< _Rep, _Period > &__d)`
- `template<typename _ToDur, typename _Clock, typename _Dur >  
constexpr enable\_if< \_\_is\_duration< _ToDur >::value, time\_point< _Clock, _ToDur > >::type time\_point\_cast  
(const time\_point< _Clock, _Dur > &__t)`

### 3.14.1 Detailed Description

ISO C++ 2011 namespace for date and time utilities.

## 3.15 std::decimal Namespace Reference

## Classes

- class [decimal128](#)
- class [decimal32](#)
- class [decimal64](#)

## Functions

- double **decimal128\_to\_double** ([decimal128](#) \_\_d)
- float **decimal128\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal128\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal128\_to\_long\_long** ([decimal128](#) \_\_d)
- double **decimal32\_to\_double** ([decimal32](#) \_\_d)
- float **decimal32\_to\_float** ([decimal32](#) \_\_d)
- long double **decimal32\_to\_long\_double** ([decimal32](#) \_\_d)
- long long **decimal32\_to\_long\_long** ([decimal32](#) \_\_d)
- double **decimal64\_to\_double** ([decimal64](#) \_\_d)
- float **decimal64\_to\_float** ([decimal64](#) \_\_d)
- long double **decimal64\_to\_long\_double** ([decimal64](#) \_\_d)
- long long **decimal64\_to\_long\_long** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal32](#) \_\_d)
- double **decimal\_to\_double** ([decimal64](#) \_\_d)
- double **decimal\_to\_double** ([decimal128](#) \_\_d)
- float **decimal\_to\_float** ([decimal32](#) \_\_d)
- float **decimal\_to\_float** ([decimal64](#) \_\_d)
- float **decimal\_to\_float** ([decimal128](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal32](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal64](#) \_\_d)
- long double **decimal\_to\_long\_double** ([decimal128](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal32](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal64](#) \_\_d)
- long long **decimal\_to\_long\_long** ([decimal128](#) \_\_d)
- static [decimal128](#) **make\_decimal128** (long long \_\_coeff, int \_\_exp)
- static [decimal128](#) **make\_decimal128** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (long long \_\_coeff, int \_\_exp)
- static [decimal32](#) **make\_decimal32** (unsigned long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (long long \_\_coeff, int \_\_exp)
- static [decimal64](#) **make\_decimal64** (unsigned long long \_\_coeff, int \_\_exp)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, long long \_\_rhs)
- bool **operator!=** ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, [decimal32](#) \_\_rhs)

- bool **operator!=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator!=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator!=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator!=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **operator\*** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **operator\*** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **operator\*** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)

- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator\\*](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator\\*](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator\\*](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator\\*](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator\\*](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator\\*](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_rhs)
- [decimal128 operator+](#) ([decimal128](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator+](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator+](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator+](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator+](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)



- **decimal64 operator+** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **decimal64 operator+** (int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator+** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_rhs)
- **decimal128 operator-** (decimal128 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, int \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- **decimal32 operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **decimal32 operator-** (int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- **decimal32 operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, int \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- **decimal64 operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)



- [decimal64 operator-](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator-](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal32](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal64](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, long long \_\_rhs)
- [decimal128 operator-](#) ([decimal128](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal128 operator-](#) (int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned int \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal128 operator-](#) (unsigned long long \_\_lhs, [decimal128](#) \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, int \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, long long \_\_rhs)
- [decimal32 operator/](#) ([decimal32](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal32 operator/](#) (int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned int \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal32 operator/](#) (unsigned long long \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal32](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, [decimal32](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, int \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, unsigned int \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, long \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, unsigned long \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, long long \_\_rhs)
- [decimal64 operator/](#) ([decimal64](#) \_\_lhs, unsigned long long \_\_rhs)
- [decimal64 operator/](#) (int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned int \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) (long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) (long long \_\_lhs, [decimal64](#) \_\_rhs)
- [decimal64 operator/](#) (unsigned long long \_\_lhs, [decimal64](#) \_\_rhs)

- **decimal128 operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long \_\_rhs)
- **decimal128 operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- **decimal128 operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **decimal128 operator/** (int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **decimal128 operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, int \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- **bool operator<** (int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, int \_\_rhs)
- **bool operator<** (int \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, long long \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, long \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- **bool operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- **bool operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- **bool operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

- bool **operator**< (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**< (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**< (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator**== (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator**== (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator**== (decimal128 \_\_lhs, unsigned long \_\_rhs)

- bool **operator==** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator==** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator>** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator>** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator>=** (long long \_\_lhs, decimal32 \_\_rhs)

- bool **operator>=** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, int \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (long \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **operator>=** (int \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, long \_\_rhs)
- bool **operator>=** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>=** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, long \_\_rhs)
- bool **operator>=** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal64 \_\_lhs, int \_\_rhs)
- bool **operator>=** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (int \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, int \_\_rhs)
- bool **operator>=** (int \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **operator>=** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **operator>=** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, long \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **operator>=** (long \_\_lhs, decimal128 \_\_rhs)
- bool **operator>=** (unsigned long long \_\_lhs, decimal128 \_\_rhs)

### 3.15.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

### 3.15.2 Function Documentation

#### 3.15.2.1 decimal32\_to\_long\_long()

```
long long std::decimal::decimal32_to_long_long (
 decimal32 __d)
```

Non-conforming extension: Conversion to integral type.

## 3.16 std::experimental Namespace Reference

### 3.16.1 Detailed Description

Namespace for features defined in ISO Technical Specifications.

## 3.17 std::literals::chrono\_literals Namespace Reference

### Functions

- constexpr [chrono::duration](#)< long double, [ratio](#)< 3600, 1 > > [operator""h](#) (long double \_\_hours)
- template<char... \_Digits>  
constexpr [chrono::hours](#) [operator""h](#) ()
- constexpr [chrono::duration](#)< long double, [ratio](#)< 60, 1 > > [operator""min](#) (long double \_\_mins)
- template<char... \_Digits>  
constexpr [chrono::minutes](#) [operator""min](#) ()
- constexpr [chrono::duration](#)< long double, [milli](#) > [operator""ms](#) (long double \_\_msecs)
- template<char... \_Digits>  
constexpr [chrono::milliseconds](#) [operator""ms](#) ()
- constexpr [chrono::duration](#)< long double, [nano](#) > [operator""ns](#) (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr [chrono::nanoseconds](#) [operator""ns](#) ()
- constexpr [chrono::duration](#)< long double > [operator""s](#) (long double \_\_secs)
- template<char... \_Digits>  
constexpr [chrono::seconds](#) [operator""s](#) ()
- constexpr [chrono::duration](#)< long double, [micro](#) > [operator""us](#) (long double \_\_usecs)
- template<char... \_Digits>  
constexpr [chrono::microseconds](#) [operator""us](#) ()

### 3.17.1 Detailed Description

ISO C++ 2014 namespace for suffixes for duration literals.

These suffixes can be used to create `chrono::duration` values with tick periods of hours, minutes, seconds, milliseconds, microseconds or nanoseconds. For example, `std::chrono::seconds(5)` can be written as `5s` after making the suffix visible in the current scope. The suffixes can be made visible by a `using-directive` or `using-declaration` such as:

- `using namespace std::chrono_literals;`
- `using namespace std::literals;`
- `using namespace std::chrono;`
- `using namespace std;`
- `using std::chrono_literals::operator""s;`

The result of these suffixes on an integer literal is one of the standard typedefs such as `std::chrono::hours`. The result on a floating-point literal is a duration type with the specified tick period and an unspecified floating-point representation, for example `1.5e2ms` might be equivalent to `chrono::duration<long double, chrono::milli>(1.5e2)`.

### 3.17.2 Function Documentation

#### 3.17.2.1 operator""h() [1/2]

```
constexpr chrono::duration<long double, ratio<3600,1> > std::literals::chrono_literals::operator""h
(
 long double __hours)
```

Literal suffix for durations representing non-integer hours.

Definition at line 1180 of file `chrono`.

#### 3.17.2.2 operator""h() [2/2]

```
template<char... _Digits>
constexpr chrono::hours std::literals::chrono_literals::operator""h ()
```

Literal suffix for durations of type `std::chrono::hours`

Definition at line 1186 of file `chrono`.

### 3.17.2.3 `operator""min()` [1/2]

```
constexpr chrono::duration<long double, ratio<60,1> > std::literals::chrono_literals::operator""min
(
 long double __mins)
```

Literal suffix for durations representing non-integer minutes.

Definition at line 1191 of file chrono.

### 3.17.2.4 `operator""min()` [2/2]

```
template<char... _Digits>
constexpr chrono::minutes std::literals::chrono_literals::operator""min ()
```

Literal suffix for durations of type `std::chrono::minutes`

Definition at line 1197 of file chrono.

### 3.17.2.5 `operator""ms()` [1/2]

```
constexpr chrono::duration<long double, milli> std::literals::chrono_literals::operator""ms (
 long double __msecs)
```

Literal suffix for durations representing non-integer milliseconds.

Definition at line 1213 of file chrono.

### 3.17.2.6 `operator""ms()` [2/2]

```
template<char... _Digits>
constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()
```

Literal suffix for durations of type `std::chrono::milliseconds`

Definition at line 1219 of file chrono.

### 3.17.2.7 `operator""ns()` [1/2]

```
constexpr chrono::duration<long double, nano> std::literals::chrono_literals::operator""ns (
 long double __nsecs)
```

Literal suffix for durations representing non-integer nanoseconds.

Definition at line 1235 of file chrono.



### 3.17.2.8 operator""ns() [2/2]

```
template<char... _Digits>
constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()
```

Literal suffix for durations of type `std::chrono::nanoseconds`

Definition at line 1241 of file `chrono`.

### 3.17.2.9 operator""s() [1/2]

```
constexpr chrono::duration<long double> std::literals::chrono_literals::operator""s (
 long double __secs)
```

Literal suffix for durations representing non-integer seconds.

Definition at line 1202 of file `chrono`.

### 3.17.2.10 operator""s() [2/2]

```
template<char... _Digits>
constexpr chrono::seconds std::literals::chrono_literals::operator""s ()
```

Literal suffix for durations of type `std::chrono::seconds`

Definition at line 1208 of file `chrono`.

### 3.17.2.11 operator""us() [1/2]

```
constexpr chrono::duration<long double, micro> std::literals::chrono_literals::operator""us (
 long double __usecs)
```

Literal suffix for durations representing non-integer microseconds.

Definition at line 1224 of file `chrono`.

### 3.17.2.12 operator""us() [2/2]

```
template<char... _Digits>
constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()
```

Literal suffix for durations of type `std::chrono::microseconds`

Definition at line 1230 of file `chrono`.

### 3.18 `std::placeholders` Namespace Reference

#### Variables

- `const _Placeholder< 1 > _1`
- `const _Placeholder< 10 > _10`
- `const _Placeholder< 11 > _11`
- `const _Placeholder< 12 > _12`
- `const _Placeholder< 13 > _13`
- `const _Placeholder< 14 > _14`
- `const _Placeholder< 15 > _15`
- `const _Placeholder< 16 > _16`
- `const _Placeholder< 17 > _17`
- `const _Placeholder< 18 > _18`
- `const _Placeholder< 19 > _19`
- `const _Placeholder< 2 > _2`
- `const _Placeholder< 20 > _20`
- `const _Placeholder< 21 > _21`
- `const _Placeholder< 22 > _22`
- `const _Placeholder< 23 > _23`
- `const _Placeholder< 24 > _24`
- `const _Placeholder< 25 > _25`
- `const _Placeholder< 26 > _26`
- `const _Placeholder< 27 > _27`
- `const _Placeholder< 28 > _28`
- `const _Placeholder< 29 > _29`
- `const _Placeholder< 3 > _3`
- `const _Placeholder< 4 > _4`
- `const _Placeholder< 5 > _5`
- `const _Placeholder< 6 > _6`
- `const _Placeholder< 7 > _7`
- `const _Placeholder< 8 > _8`
- `const _Placeholder< 9 > _9`

#### 3.18.1 Detailed Description

ISO C++ 2011 namespace for `std::bind` placeholders.

### 3.19 `std::regex_constants` Namespace Reference

#### 5.1 Regular Expression Syntax Options

- `enum __syntax_option {  
    _S_icode, _S_nosubs, _S_optimize, _S_collate,  
    _S_ECMAScript, _S_basic, _S_extended, _S_awk,  
    _S_grep, _S_egrep, _S_polynomial, _S_syntax_last }`
- `enum syntax_option_type` : unsigned int
- `constexpr syntax_option_type icode`

- constexpr [syntax\\_option\\_type](#) nosubs
- constexpr [syntax\\_option\\_type](#) optimize
- constexpr [syntax\\_option\\_type](#) collate
- constexpr [syntax\\_option\\_type](#) ECMAScript
- constexpr [syntax\\_option\\_type](#) basic
- constexpr [syntax\\_option\\_type](#) extended
- constexpr [syntax\\_option\\_type](#) awk
- constexpr [syntax\\_option\\_type](#) grep
- constexpr [syntax\\_option\\_type](#) egrep
- constexpr [syntax\\_option\\_type](#) \_\_polynomial
- constexpr [syntax\\_option\\_type](#) operator& ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b)
- constexpr [syntax\\_option\\_type](#) operator| ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b)
- constexpr [syntax\\_option\\_type](#) operator^ ([syntax\\_option\\_type](#) \_\_a, [syntax\\_option\\_type](#) \_\_b)
- constexpr [syntax\\_option\\_type](#) operator~ ([syntax\\_option\\_type](#) \_\_a)
- [syntax\\_option\\_type](#) & operator&= ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b)
- [syntax\\_option\\_type](#) & operator|= ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b)
- [syntax\\_option\\_type](#) & operator^= ([syntax\\_option\\_type](#) &\_\_a, [syntax\\_option\\_type](#) \_\_b)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [\\_\\_match\\_flag](#) {  
[\\_S\\_not\\_bol](#), [\\_S\\_not\\_eol](#), [\\_S\\_not\\_bow](#), [\\_S\\_not\\_eow](#),  
[\\_S\\_any](#), [\\_S\\_not\\_null](#), [\\_S\\_continuous](#), [\\_S\\_prev\\_avail](#),  
[\\_S\\_sed](#), [\\_S\\_no\\_copy](#), [\\_S\\_first\\_only](#), [\\_S\\_match\\_flag\\_last](#) }
- enum [match\\_flag\\_type](#) : unsigned int
- constexpr [match\\_flag\\_type](#) match\_default
- constexpr [match\\_flag\\_type](#) match\_not\_bol
- constexpr [match\\_flag\\_type](#) match\_not\_eol
- constexpr [match\\_flag\\_type](#) match\_not\_bow
- constexpr [match\\_flag\\_type](#) match\_not\_eow
- constexpr [match\\_flag\\_type](#) match\_any
- constexpr [match\\_flag\\_type](#) match\_not\_null
- constexpr [match\\_flag\\_type](#) match\_continuous
- constexpr [match\\_flag\\_type](#) match\_prev\_avail
- constexpr [match\\_flag\\_type](#) format\_default
- constexpr [match\\_flag\\_type](#) format\_sed
- constexpr [match\\_flag\\_type](#) format\_no\_copy
- constexpr [match\\_flag\\_type](#) format\_first\_only
- constexpr [match\\_flag\\_type](#) operator& ([match\\_flag\\_type](#) \_\_a, [match\\_flag\\_type](#) \_\_b)
- constexpr [match\\_flag\\_type](#) operator| ([match\\_flag\\_type](#) \_\_a, [match\\_flag\\_type](#) \_\_b)
- constexpr [match\\_flag\\_type](#) operator^ ([match\\_flag\\_type](#) \_\_a, [match\\_flag\\_type](#) \_\_b)
- constexpr [match\\_flag\\_type](#) operator~ ([match\\_flag\\_type](#) \_\_a)
- [match\\_flag\\_type](#) & operator&= ([match\\_flag\\_type](#) &\_\_a, [match\\_flag\\_type](#) \_\_b)
- [match\\_flag\\_type](#) & operator|= ([match\\_flag\\_type](#) &\_\_a, [match\\_flag\\_type](#) \_\_b)
- [match\\_flag\\_type](#) & operator^= ([match\\_flag\\_type](#) &\_\_a, [match\\_flag\\_type](#) \_\_b)

### 5.3 Error Types

- enum `error_type` {  
    `_S_error_collate`, `_S_error_ctype`, `_S_error_escape`, `_S_error_backref`,  
    `_S_error_brack`, `_S_error_paren`, `_S_error_brace`, `_S_error_badbrace`,  
    `_S_error_range`, `_S_error_space`, `_S_error_badrepeat`, `_S_error_complexity`,  
    `_S_error_stack` }
- constexpr `error_type error_collate` (`_S_error_collate`)
- constexpr `error_type error_ctype` (`_S_error_ctype`)
- constexpr `error_type error_escape` (`_S_error_escape`)
- constexpr `error_type error_backref` (`_S_error_backref`)
- constexpr `error_type error_brack` (`_S_error_brack`)
- constexpr `error_type error_paren` (`_S_error_paren`)
- constexpr `error_type error_brace` (`_S_error_brace`)
- constexpr `error_type error_badbrace` (`_S_error_badbrace`)
- constexpr `error_type error_range` (`_S_error_range`)
- constexpr `error_type error_space` (`_S_error_space`)
- constexpr `error_type error_badrepeat` (`_S_error_badrepeat`)
- constexpr `error_type error_complexity` (`_S_error_complexity`)
- constexpr `error_type error_stack` (`_S_error_stack`)

#### 3.19.1 Detailed Description

ISO C++ 2011 namespace for options and flags used with `std::regex`.

#### 3.19.2 Enumeration Type Documentation

##### 3.19.2.1 `__match_flag`

```
enum std::regex_constants::__match_flag
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 232 of file `regex_constants.h`.

### 3.19.2.2 \_\_syntax\_option

```
enum std::regex_constants::__syntax_option
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 54 of file `regex_constants.h`.

### 3.19.2.3 error\_type

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

Definition at line 49 of file `regex_error.h`.

### 3.19.2.4 match\_flag\_type

```
enum std::regex_constants::match_flag_type : unsigned int
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 255 of file `regex_constants.h`.

### 3.19.2.5 syntax\_option\_type

```
enum std::regex_constants::syntax_option_type : unsigned int
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 81 of file `regex_constants.h`.

### 3.19.3 Function Documentation

#### 3.19.3.1 `error_backref()`

```
constexpr error_type std::regex_constants::error_backref (
 _S_error_backref)
```

The expression contained an invalid back reference.

#### 3.19.3.2 `error_badbrace()`

```
constexpr error_type std::regex_constants::error_badbrace (
 _S_error_badbrace)
```

The expression contained an invalid range in a {} expression.

#### 3.19.3.3 `error_badrepeat()`

```
constexpr error_type std::regex_constants::error_badrepeat (
 _S_error_badrepeat)
```

One of \*?+{ was not preceded by a valid regular expression.

#### 3.19.3.4 `error_brace()`

```
constexpr error_type std::regex_constants::error_brace (
 _S_error_brace)
```

The expression contained mismatched { and }

#### 3.19.3.5 `error_brack()`

```
constexpr error_type std::regex_constants::error_brack (
 _S_error_brack)
```

The expression contained mismatched [ and ].

#### 3.19.3.6 `error_collate()`

```
constexpr error_type std::regex_constants::error_collate (
 _S_error_collate)
```

The expression contained an invalid collating element name.

#### 3.19.3.7 error\_complexity()

```
constexpr error_type std::regex_constants::error_complexity (
 _S_error_complexity)
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

#### 3.19.3.8 error\_ctype()

```
constexpr error_type std::regex_constants::error_ctype (
 _S_error_ctype)
```

The expression contained an invalid character class name.

#### 3.19.3.9 error\_escape()

```
constexpr error_type std::regex_constants::error_escape (
 _S_error_escape)
```

The expression contained an invalid escaped character, or a trailing escape.

#### 3.19.3.10 error\_paren()

```
constexpr error_type std::regex_constants::error_paren (
 _S_error_paren)
```

The expression contained mismatched ( and ).

#### 3.19.3.11 error\_range()

```
constexpr error_type std::regex_constants::error_range (
 _S_error_range)
```

The expression contained an invalid character range, such as [b-a] in most encodings.

#### 3.19.3.12 error\_space()

```
constexpr error_type std::regex_constants::error_space (
 _S_error_space)
```

There was insufficient memory to convert the expression into a finite state machine.

#### 3.19.3.13 error\_stack()

```
constexpr error_type std::regex_constants::error_stack (
 _S_error_stack)
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

**3.19.3.14 operator&()** [1/2]

```
constexpr syntax_option_type std::regex_constants::operator& (
 syntax_option_type __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 183 of file `regex_constants.h`.

**3.19.3.15 operator&()** [2/2]

```
constexpr match_flag_type std::regex_constants::operator& (
 match_flag_type __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 374 of file `regex_constants.h`.

**3.19.3.16 operator&=()** [1/2]

```
syntax_option_type& std::regex_constants::operator&= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 208 of file `regex_constants.h`.



### 3.19.3.17 operator&=() [2/2]

```
match_flag_type& std::regex_constants::operator&= (
 match_flag_type & __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 399 of file `regex_constants.h`.

### 3.19.3.18 operator^() [1/2]

```
constexpr syntax_option_type std::regex_constants::operator^ (
 syntax_option_type __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 197 of file `regex_constants.h`.

### 3.19.3.19 operator^() [2/2]

```
constexpr match_flag_type std::regex_constants::operator^ (
 match_flag_type __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 388 of file `regex_constants.h`.

**3.19.3.20 operator^=()** [1/2]

```
syntax_option_type& std::regex_constants::operator^= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 216 of file `regex_constants.h`.

**3.19.3.21 operator^=()** [2/2]

```
match_flag_type& std::regex_constants::operator^= (
 match_flag_type & __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 407 of file `regex_constants.h`.

**3.19.3.22 operator" | ()** [1/2]

```
constexpr syntax_option_type std::regex_constants::operator| (
 syntax_option_type __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 190 of file `regex_constants.h`.

### 3.19.3.23 operator" | () [2/2]

```
constexpr match_flag_type std::regex_constants::operator| (
 match_flag_type __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 381 of file `regex_constants.h`.

### 3.19.3.24 operator" |=() [1/2]

```
syntax_option_type& std::regex_constants::operator|= (
 syntax_option_type & __a,
 syntax_option_type __b) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 212 of file `regex_constants.h`.

### 3.19.3.25 operator" |=() [2/2]

```
match_flag_type& std::regex_constants::operator|= (
 match_flag_type & __a,
 match_flag_type __b) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 403 of file `regex_constants.h`.

### 3.19.3.26 `operator~()` [1/2]

```
constexpr syntax_option_type std::regex_constants::operator~ (
 syntax_option_type __a) [inline]
```

This is a bitmask type indicating how to interpret the regex.

The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type `syntax_option_type` shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

Definition at line 204 of file `regex_constants.h`.

### 3.19.3.27 `operator~()` [2/2]

```
constexpr match_flag_type std::regex_constants::operator~ (
 match_flag_type __a) [inline]
```

This is a bitmask type indicating regex matching rules.

The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

Definition at line 395 of file `regex_constants.h`.

## 3.19.4 Variable Documentation

### 3.19.4.1 `__polynomial`

```
constexpr syntax_option_type std::regex_constants::__polynomial [inline]
```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception `regex_constants::error_complexity` will be thrown.

Definition at line 179 of file `regex_constants.h`.

### 3.19.4.2 `awk`

```
constexpr syntax_option_type std::regex_constants::awk [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `awk` in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` `extended`, except that C-style escape sequences are supported. These sequences are: `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`, `\&apos;`, `&apos;`, and `\ddd` (where `ddd` is one, two, or three octal digits).

Definition at line 152 of file `regex_constants.h`.

#### 3.19.4.3 basic

```
constexpr syntax_option_type std::regex_constants::basic [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 132 of file `regex_constants.h`.

#### 3.19.4.4 collate

```
constexpr syntax_option_type std::regex_constants::collate [inline]
```

Specifies that character ranges of the form `[a-b]` should be locale sensitive.

Definition at line 111 of file `regex_constants.h`.

#### 3.19.4.5 ECMAScript

```
constexpr syntax_option_type std::regex_constants::ECMAScript [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

Definition at line 122 of file `regex_constants.h`.

#### 3.19.4.6 egrep

```
constexpr syntax_option_type std::regex_constants::egrep [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility `grep` when given the `-E` option in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type` extended, except that newlines are treated as whitespace.

Definition at line 170 of file `regex_constants.h`.

### 3.19.4.7 extended

```
constexpr syntax_option_type std::regex_constants::extended [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

Definition at line 141 of file `regex_constants.h`.

### 3.19.4.8 format\_default

```
constexpr match_flag_type std::regex_constants::format_default [inline]
```

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.

Format strings (from ECMA-262 [15.5.4.11]):

- `$$` The dollar-sign itself (`$`).
- `$&` The matched substring.
- `$`` The portion of *string* that precedes the matched substring. This would be `match_results::prefix()`.
- `$'` The portion of *string* that follows the matched substring. This would be `match_results::suffix()`.
- `$n` The *n*th capture, where *n* is in [1,9] and `$n` is not followed by a decimal digit. If `n <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.
- `$nn` The *nn*th capture, where *nn* is a two-digit decimal number on [01, 99]. If `nn <= match_results::size()` and the *n*th capture is undefined, use the empty string instead. If `nn > match_results::size()`, the result is implementation-defined.

Definition at line 346 of file `regex_constants.h`.

### 3.19.4.9 format\_first\_only

```
constexpr match_flag_type std::regex_constants::format_first_only [inline]
```

When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

Definition at line 370 of file `regex_constants.h`.

#### 3.19.4.10 format\_no\_copy

```
constexpr match_flag_type std::regex_constants::format_no_copy [inline]
```

During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

Definition at line 363 of file `regex_constants.h`.

#### 3.19.4.11 format\_sed

```
constexpr match_flag_type std::regex_constants::format_sed [inline]
```

When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

Definition at line 355 of file `regex_constants.h`.

#### 3.19.4.12 grep

```
constexpr syntax_option_type std::regex_constants::grep [inline]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to `syntax_option_type basic`, except that newlines are treated as whitespace.

Definition at line 161 of file `regex_constants.h`.

#### 3.19.4.13 icase

```
constexpr syntax_option_type std::regex_constants::icase [inline]
```

Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

Definition at line 87 of file `regex_constants.h`.

#### 3.19.4.14 match\_any

```
constexpr match_flag_type std::regex_constants::match_any [inline]
```

If more than one match is possible then any match is an acceptable result.

Definition at line 297 of file `regex_constants.h`.

#### 3.19.4.15 `match_continuous`

```
constexpr match_flag_type std::regex_constants::match_continuous [inline]
```

The expression only matches a sub-sequence that begins at first .

Definition at line 309 of file `regex_constants.h`.

#### 3.19.4.16 `match_default`

```
constexpr match_flag_type std::regex_constants::match_default [inline]
```

The default matching rules.

Definition at line 260 of file `regex_constants.h`.

#### 3.19.4.17 `match_not_bol`

```
constexpr match_flag_type std::regex_constants::match_not_bol [inline]
```

The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

Definition at line 268 of file `regex_constants.h`.

#### 3.19.4.18 `match_not_bow`

```
constexpr match_flag_type std::regex_constants::match_not_bow [inline]
```

The expression \b is not matched against the sub-sequence [first,first).

Definition at line 283 of file `regex_constants.h`.

#### 3.19.4.19 `match_not_eol`

```
constexpr match_flag_type std::regex_constants::match_not_eol [inline]
```

The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character (\$) in the regular expression shall not match [last, last).

Definition at line 276 of file `regex_constants.h`.



#### 3.19.4.20 match\_not\_eow

```
constexpr match_flag_type std::regex_constants::match_not_eow [inline]
```

The expression \b should not be matched against the sub-sequence [last,last).

Definition at line 290 of file regex\_constants.h.

#### 3.19.4.21 match\_not\_null

```
constexpr match_flag_type std::regex_constants::match_not_null [inline]
```

The expression does not match an empty sequence.

Definition at line 303 of file regex\_constants.h.

#### 3.19.4.22 match\_prev\_avail

```
constexpr match_flag_type std::regex_constants::match_prev_avail [inline]
```

—first is a valid iterator position. When this flag is set then the flags match\_not\_bol and match\_not\_bow are ignored by the regular expression algorithms 28.11 and iterators 28.12.

Definition at line 317 of file regex\_constants.h.

#### 3.19.4.23 nosubs

```
constexpr syntax_option_type std::regex_constants::nosubs [inline]
```

Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied match\_results structure.

Definition at line 95 of file regex\_constants.h.

#### 3.19.4.24 optimize

```
constexpr syntax_option_type std::regex_constants::optimize [inline]
```

Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

Definition at line 104 of file regex\_constants.h.

## 3.20 std::rel\_ops Namespace Reference

### Functions

- `template<class _Tp >`  
`bool operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool operator>= (const _Tp &__x, const _Tp &__y)`

### 3.20.1 Detailed Description

The generated relational operators are sequestered here.

### 3.20.2 Function Documentation

#### 3.20.2.1 operator"!=()"

```
template<class _Tp >
bool std::rel_ops::operator!= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines != for arbitrary types, in terms of ==.

#### Parameters

|                          |                |
|--------------------------|----------------|
| $\leftrightarrow$<br>__x | A thing.       |
| $\leftrightarrow$<br>__y | Another thing. |

#### Returns

`__x != __y`

This function uses == to determine its result.

Definition at line 87 of file stl\_relops.h.

### 3.20.2.2 operator<=()

```
template<class _Tp >
bool std::rel_ops::operator<= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines <= for arbitrary types, in terms of <.

#### Parameters

|       |                |
|-------|----------------|
| $\_x$ | A thing.       |
| $\_y$ | Another thing. |

#### Returns

$\_x <= \_y$

This function uses < to determine its result.

Definition at line 113 of file stl\_relops.h.

### 3.20.2.3 operator>()

```
template<class _Tp >
bool std::rel_ops::operator> (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines > for arbitrary types, in terms of <.

#### Parameters

|       |                |
|-------|----------------|
| $\_x$ | A thing.       |
| $\_y$ | Another thing. |

#### Returns

$\_x > \_y$

This function uses < to determine its result.

Definition at line 100 of file stl\_relops.h.

### 3.20.2.4 operator>=()

```
template<class _Tp >
bool std::rel_ops::operator>= (
 const _Tp & __x,
 const _Tp & __y) [inline]
```

Defines >= for arbitrary types, in terms of <.

#### Parameters

|       |                |
|-------|----------------|
| $\_x$ | A thing.       |
| $\_y$ | Another thing. |

#### Returns

$\_x \geq \_y$

This function uses < to determine its result.

Definition at line 126 of file stl\_relops.h.

## 3.21 std::this\_thread Namespace Reference

### Functions

- void **\_\_sleep\_for** ([chrono::seconds](#), [chrono::nanoseconds](#))
- [thread::id get\\_id](#) () noexcept
- template<typename \_Rep , typename \_Period >  
void [sleep\\_for](#) (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock , typename \_Duration >  
void [sleep\\_until](#) (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void [yield](#) () noexcept

#### 3.21.1 Detailed Description

ISO C++ 2011 namespace for interacting with the current thread.

C++11 30.3.2 [thread.thread.this] Namespace this\_thread.

#### 3.21.2 Function Documentation

### 3.21.2.1 get\_id()

```
thread::id std::this_thread::get_id () [inline], [noexcept]
```

get\_id

Definition at line 365 of file thread.

### 3.21.2.2 sleep\_for()

```
template<typename _Rep , typename _Period >
void std::this_thread::sleep_for (
 const chrono::duration< _Rep, _Period > & __rtime) [inline]
```

sleep\_for

Definition at line 389 of file thread.

### 3.21.2.3 sleep\_until()

```
template<typename _Clock , typename _Duration >
void std::this_thread::sleep_until (
 const chrono::time_point< _Clock, _Duration > & __atime) [inline]
```

sleep\_until

Definition at line 411 of file thread.

### 3.21.2.4 yield()

```
void std::this_thread::yield () [inline], [noexcept]
```

yield

Definition at line 376 of file thread.

## 3.22 std::tr1 Namespace Reference

Namespaces

- [\\_\\_detail](#)

## Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc\_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- `float assoc\_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- `long double assoc\_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type assoc\_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- `float assoc\_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- `long double assoc\_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- `float betaf` (float \_\_x, float \_\_y)
- `long double betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp\_ellint\_1` (\_Tp \_\_k)
- `float comp\_ellint\_1f` (float \_\_k)
- `long double comp\_ellint\_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type comp\_ellint\_2` (\_Tp \_\_k)
- `float comp\_ellint\_2f` (float \_\_k)
- `long double comp\_ellint\_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp\_ellint\_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- `float comp\_ellint\_3f` (float \_\_k, float \_\_nu)
- `long double comp\_ellint\_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf\_hyperg` (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- `float conf\_hypergf` (float \_\_a, float \_\_c, float \_\_x)
- `long double conf\_hypergl` (long double \_\_a, long double \_\_c, long double \_\_x)
- `template<typename _Tp >`  
`std::complex< _Tp > conj` (const `std::complex`< \_Tp > &\_\_z)
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > conj` (\_Tp \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float cyl\_bessel\_if` (float \_\_nu, float \_\_x)
- `long double cyl\_bessel\_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float cyl\_bessel\_jf` (float \_\_nu, float \_\_x)
- `long double cyl\_bessel\_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_bessel\_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float cyl\_bessel\_kf` (float \_\_nu, float \_\_x)
- `long double cyl\_bessel\_kl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl\_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- `float cyl\_neumannf` (float \_\_nu, float \_\_x)
- `long double cyl\_neumannl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint\_1` (\_Tp \_\_k, \_Tpp \_\_phi)

- float **ellint\_1f** (float \_\_k, float \_\_phi)
- long double **ellint\_1l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Tpp >::\_\_type **ellint\_2** (\_Tp \_\_k, \_Tpp \_\_phi)
- float **ellint\_2f** (float \_\_k, float \_\_phi)
- long double **ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **expint** (\_Tp \_\_x)
- float **expintf** (float \_\_x)
- long double **expintl** (long double \_\_x)
- template<typename \_Tp >  
**std::complex**< \_Tp > **fabs** (const **std::complex**< \_Tp > &\_\_z)
- float **fabs** (float \_\_x)
- long double **fabs** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **fabs** (\_Tp \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **hermitef** (unsigned int \_\_n, float \_\_x)
- long double **hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **laguerref** (unsigned int \_\_n, float \_\_x)
- long double **laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **legendref** (unsigned int \_\_n, float \_\_x)
- long double **legendrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp, typename \_Up >  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **polar** (const \_Tp &\_\_rho, const \_Up &\_\_theta)
- template<typename \_Tp, typename \_Up >  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const **std::complex**< \_Tp > &\_\_x, const \_Up &\_\_y)
- template<typename \_Tp, typename \_Up >  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const \_Tp &\_\_x, const **std::complex**< \_Up > &\_\_y)
- template<typename \_Tp, typename \_Up >  
**std::complex**< typename \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type > **pow** (const **std::complex**< \_Tp > &\_\_x, const **std::complex**< \_Up > &\_\_y)
- template<typename \_Tp >  
**std::complex**< \_Tp > **pow** (const **std::complex**< \_Tp > &\_\_x, const \_Tp &\_\_y)
- template<typename \_Tp >  
**std::complex**< \_Tp > **pow** (const \_Tp &\_\_x, const **std::complex**< \_Tp > &\_\_y)

- `template<typename _Tp >`  
`std::complex< _Tp > pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`
- `float pow (float __x, float __y)`
- `long double pow (long double __x, long double __y)`
- `template<typename _Tp, typename _Up >`  
`__gnu_cxx::__promote_2< _Tp, _Up >::__type pow (_Tp __x, _Up __y)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)`
- `float riemann_zetaf (float __x)`
- `long double riemann_zetal (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)`
- `float sph_besself (unsigned int __n, float __x)`
- `long double sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float sph_legendref (unsigned int __l, unsigned int __m, float __theta)`
- `long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)`
- `float sph_neumannf (unsigned int __n, float __x)`
- `long double sph_neumannl (unsigned int __n, long double __x)`

### 3.22.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is `std::tr1`.

## 3.23 `std::tr1::__detail` Namespace Reference

### 3.23.1 Detailed Description

Implementation details not part of the namespace `std::tr1` interface.

## 3.24 `std::tr2` Namespace Reference

### Namespaces

- [`\_\_detail`](#)

### Classes

- `struct __dynamic_bitset_base`
- `struct __reflection_typelist`
- `struct __reflection_typelist< _First, _Rest... >`
- `struct __reflection_typelist<>`
- `struct bases`
- `class bool_set`
- `struct direct_bases`
- `class dynamic_bitset`



## Functions

- bool **certainly** (bool\_set \_\_b)
- bool **contains** (bool\_set \_\_s, bool\_set \_\_t)
- bool **equals** (bool\_set \_\_s, bool\_set \_\_t)
- bool **is\_emptyset** (bool\_set \_\_b)
- bool **is\_indeterminate** (bool\_set \_\_b)
- bool **is\_singleton** (bool\_set \_\_b)
- bool\_set **operator!=** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator!=** (bool\_set \_\_s, bool \_\_t)
- bool\_set **operator!=** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **operator&** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator&** (bool\_set \_\_s, bool \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc >  
std::basic\_ostream< \_CharT, \_Traits > & **operator<<** (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const  
dynamic\_bitset< \_WordT, \_Alloc > &\_\_x)
- bool\_set **operator==** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator==** (bool\_set \_\_s, bool \_\_t)
- template<typename \_CharT, typename \_Traits, typename \_WordT, typename \_Alloc >  
std::basic\_istream< \_CharT, \_Traits > & **operator>>** (std::basic\_istream< \_CharT, \_Traits > &\_\_is,  
dynamic\_bitset< \_WordT, \_Alloc > &\_\_x)
- bool\_set **operator^** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator^** (bool\_set \_\_s, bool \_\_t)
- bool\_set **operator|** (bool \_\_s, bool\_set \_\_t)
- bool\_set **operator|** (bool\_set \_\_s, bool \_\_t)
- bool **possibly** (bool\_set \_\_b)
- bool\_set **set\_complement** (bool\_set \_\_b)
- bool\_set **set\_intersection** (bool \_\_s, bool\_set \_\_t)
- bool\_set **set\_intersection** (bool\_set \_\_s, bool \_\_t)
- bool\_set **set\_intersection** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **set\_union** (bool \_\_s, bool\_set \_\_t)
- bool\_set **set\_union** (bool\_set \_\_s, bool \_\_t)
- bool\_set **set\_union** (bool\_set \_\_s, bool\_set \_\_t)
- template<typename \_WordT, typename \_Alloc >  
bool **operator!=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator<=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator>** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)
- template<typename \_WordT, typename \_Alloc >  
bool **operator>=** (const dynamic\_bitset< \_WordT, \_Alloc > &\_\_lhs, const dynamic\_bitset< \_WordT, \_Alloc >  
&\_\_rhs)

- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const`  
`dynamic_bitset< _WordT, _Alloc > &__y)`

#### 3.24.1 Detailed Description

Namespace for non-standard "TR2" extensions.

### 3.25 std::tr2::\_\_detail Namespace Reference

#### 3.25.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

## 4 Class Documentation

### 4.1 \_\_gnu\_parallel::\_\_accumulate\_binop\_reduct< \_BinOp > Struct Template Reference

#### Public Member Functions

- `__accumulate_binop_reduct (_BinOp &__b)`
- `template<typename _Result, typename _Addend >`  
`_Result operator() (const _Result &__x, const _Addend &__y)`

#### Public Attributes

- `_BinOp & __binop`

#### 4.1.1 Detailed Description

```
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >
```

General reduction, using a binary operator.

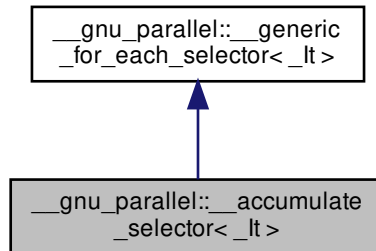
Definition at line 335 of file for\_each\_selectors.h.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

4.2 `__gnu_parallel::__accumulate_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__accumulate_selector<_It>`:



## Public Member Functions

- `template<typename _Op>`  
`std::iterator_traits<_It>::value_type operator() (_Op __o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.2.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__accumulate_selector<_It>
```

`std::accumulate()` selector.

Definition at line 208 of file `for_each_selectors.h`.

## 4.2.2 Member Function Documentation

4.2.2.1 `operator()`

```
template<typename _It>
template<typename _Op>
std::iterator_traits<_It>::value_type __gnu_parallel::__accumulate_selector<_It>::operator() (
 _Op __o,
 _It __i) [inline]
```

Functor execution.

## Parameters

|                        |                              |
|------------------------|------------------------------|
| <code>_↔<br/>_o</code> | Operator (unused).           |
| <code>_↔<br/>_i</code> | iterator referencing object. |

## Returns

The current value.

Definition at line 216 of file `for_each_selectors.h`.

## 4.2.3 Member Data Documentation

4.2.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

4.3 `std::__add_pointer_helper< _Tp, bool >` Struct Template Reference

## Public Types

- `typedef _Tp type`

## 4.3.1 Detailed Description

```
template<typename _Tp, bool = __or_<__is_referenceable<_Tp>, is_void<_Tp>>::value>
struct std::__add_pointer_helper< _Tp, bool >
```

`add_pointer`

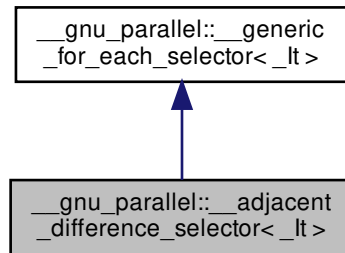
Definition at line 2025 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.4 `__gnu_parallel::__adjacent_difference_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__adjacent_difference_selector<_It>`:



## Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It \_M\_finish\_iterator`

## 4.4.1 Detailed Description

```

template<typename _It>
struct __gnu_parallel::__adjacent_difference_selector<_It>

```

Selector that returns the difference between two adjacent `__elements`.

Definition at line 269 of file `for_each_selectors.h`.

## 4.4.2 Member Data Documentation

#### 4.4.2.1 `_M_finish_iterator`

```
template<typename _It >
__It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

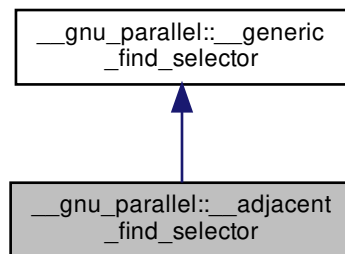
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.5 `__gnu_parallel::__adjacent_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__adjacent_find_selector`:



#### Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator ( _RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

##### 4.5.1 Detailed Description

Test predicate on two adjacent elements.

Definition at line 80 of file `find_selectors.h`.

## 4.5.2 Member Function Documentation

## 4.5.2.1 \_M\_sequential\_algorithm()

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

## Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

Definition at line 105 of file `find_selectors.h`.

## 4.5.2.2 operator()()

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__adjacent_find_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

## Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

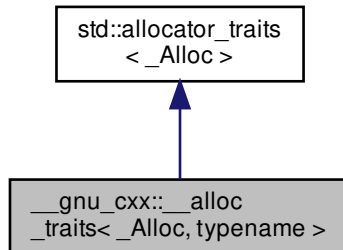
Definition at line 90 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

#### 4.6 `__gnu_cxx::__alloc_traits<_Alloc, typename >` Struct Template Reference

Inheritance diagram for `__gnu_cxx::__alloc_traits<_Alloc, typename >`:



##### Public Types

- typedef `std::allocator_traits<_Alloc>` **\_Base\_type**
- typedef `_Alloc` **allocator\_type**
- typedef `_Base_type::const_pointer` **const\_pointer**
- typedef `const value_type & const_reference`
- using `const_void_pointer` = `typename _Ptr<__cv_pointer, const void>::type`
- typedef `_Base_type::difference_type` **difference\_type**
- using `is_always_equal` = `__detected_or_t<typename is_empty<_Alloc>::type, __equal, _Alloc>`
- typedef `_Base_type::pointer` **pointer**
- using `propagate_on_container_copy_assignment` = `__detected_or_t<false_type, __pocca, _Alloc>`
- using `propagate_on_container_move_assignment` = `__detected_or_t<false_type, __pocma, _Alloc>`
- using `propagate_on_container_swap` = `__detected_or_t<false_type, __pocs, _Alloc>`
- template<typename `_Tp`>  
using **rebind\_alloc** = `__alloc_rebind<_Alloc, _Tp>`
- template<typename `_Tp`>  
using **rebind\_traits** = `allocator_traits<rebind_alloc<_Tp>>`
- typedef `value_type & reference`
- typedef `_Base_type::size_type` **size\_type**
- typedef `_Base_type::value_type` **value\_type**
- using `void_pointer` = `typename _Ptr<__v_pointer, void>::type`

##### Static Public Member Functions

- static constexpr bool **\_S\_always\_equal** ()
- static constexpr bool **\_S\_nothrow\_move** ()
- static constexpr void **\_S\_on\_swap** (`_Alloc &_a, _Alloc &_b`)
- static constexpr bool **\_S\_propagate\_on\_copy\_assign** ()
- static constexpr bool **\_S\_propagate\_on\_move\_assign** ()



- static constexpr bool **\_S\_propagate\_on\_swap** ()
- static constexpr `_Alloc` **\_S\_select\_on\_copy** (const `_Alloc` &\_\_a)
- static constexpr `pointer allocate` (`_Alloc` &\_\_a, `size_type` \_\_n)
- static constexpr `pointer allocate` (`_Alloc` &\_\_a, `size_type` \_\_n, `const_void_pointer` \_\_hint)
- static constexpr `pointer allocate` (`_Alloc` &\_\_a, `size_type` \_\_n)
- static constexpr `pointer allocate` (`_Alloc` &\_\_a, `size_type` \_\_n, `const_void_pointer` \_\_hint)
- template<typename `_Tp`, typename... `_Args`>  
static constexpr auto **construct** (`_Alloc` &\_\_a, `_Tp` \*\_\_p, `_Args` &&... \_\_args) noexcept(noexcept(`_S_construct`(`__a`, \_\_p, `std::forward`<`_Args`>(\_\_args)...))) -> decltype(`_S_construct`(`__a`, \_\_p, `std::forward`<`_Args`>(\_\_args)...))
- template<typename `_Ptr`, typename... `_Args`>  
static constexpr std::enable\_if\_t<\_\_is\_custom\_pointer<`_Ptr`>::value > **construct** (`_Alloc` &\_\_a, `_Ptr` \_\_p, `_Args` &&... \_\_args) noexcept(noexcept(`_Base_type::construct`(`__a`, std::to\_address(\_\_p), `std::forward`<`_Args`>(\_\_args)...)))
- template<typename `_Tp`, typename... `_Args`>  
static constexpr auto **construct** (`_Alloc` &\_\_a, `_Tp` \*\_\_p, `_Args` &&... \_\_args) noexcept(noexcept(`_S_construct`(`__a`, \_\_p, `std::forward`<`_Args`>(\_\_args)...))) -> decltype(`_S_construct`(`__a`, \_\_p, `std::forward`<`_Args`>(\_\_args)...))
- static constexpr void **deallocate** (`_Alloc` &\_\_a, `pointer` \_\_p, `size_type` \_\_n)
- static constexpr void **deallocate** (`_Alloc` &\_\_a, `pointer` \_\_p, `size_type` \_\_n)
- template<typename `_Tp`>  
static constexpr void **destroy** (`_Alloc` &\_\_a, `_Tp` \*\_\_p) noexcept(noexcept(`_S_destroy`(`__a`, \_\_p, 0)))
- template<typename `_Ptr`>  
static constexpr std::enable\_if\_t<\_\_is\_custom\_pointer<`_Ptr`>::value > **destroy** (`_Alloc` &\_\_a, `_Ptr` \_\_p) noexcept(noexcept(`_Base_type::destroy`(`__a`, std::to\_address(\_\_p))))
- template<typename `_Tp`>  
static constexpr void **destroy** (`_Alloc` &\_\_a, `_Tp` \*\_\_p) noexcept(noexcept(`_S_destroy`(`__a`, \_\_p, 0)))
- static constexpr `size_type max_size` (const `_Alloc` &\_\_a) noexcept
- static constexpr `size_type max_size` (const `_Alloc` &\_\_a) noexcept
- static constexpr `_Alloc select_on_container_copy_construction` (const `_Alloc` &\_\_rhs)

### Protected Types

- template<typename `_Tp`>  
using **\_\_c\_pointer** = typename `_Tp::const_pointer`
- template<typename `_Tp`>  
using **\_\_cv\_pointer** = typename `_Tp::const_void_pointer`
- template<typename `_Tp`>  
using **\_\_equal** = typename `_Tp::is_always_equal`
- template<typename `_Tp`>  
using **\_\_pocca** = typename `_Tp::propagate_on_container_copy_assignment`
- template<typename `_Tp`>  
using **\_\_pocma** = typename `_Tp::propagate_on_container_move_assignment`
- template<typename `_Tp`>  
using **\_\_pocs** = typename `_Tp::propagate_on_container_swap`
- template<typename `_Tp`>  
using **\_\_pointer** = typename `_Tp::pointer`
- template<typename `_Tp`>  
using **\_\_v\_pointer** = typename `_Tp::void_pointer`

#### 4.6.1 Detailed Description

```
template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits< _Alloc, typename >
```

Uniform interface to C++98 and C++11 allocators.

Definition at line 48 of file ext/alloc\_traits.h.

#### 4.6.2 Member Typedef Documentation

##### 4.6.2.1 const\_void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer = typename _Ptr<__cv_pointer, const
void>::type [inherited]
```

The allocator's const void pointer type.

`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

Definition at line 154 of file bits/alloc\_traits.h.

##### 4.6.2.2 is\_always\_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal = __detected_or_t<typename is_empty<_↵
Alloc>::type, __equal, _Alloc> [inherited]
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

Definition at line 206 of file bits/alloc\_traits.h.

##### 4.6.2.3 propagate\_on\_container\_copy\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment = __detected_or_↵
t<false_type, __pocca, _Alloc> [inherited]
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 179 of file bits/alloc\_traits.h.

4.6.2.4 `propagate_on_container_move_assignment`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment = __detected_or_t<
false_type, __pocma, _Alloc> [inherited]
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file `bits/alloc_traits.h`.

4.6.2.5 `propagate_on_container_swap`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap = __detected_or_t<false_type,
__pocs, _Alloc> [inherited]
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

Definition at line 197 of file `bits/alloc_traits.h`.

4.6.2.6 `void_pointer`

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer = typename _Ptr<__v_pointer, void>::type
[inherited]
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

Definition at line 146 of file `bits/alloc_traits.h`.

## 4.6.3 Member Function Documentation

4.6.3.1 `allocate()` [1/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator_traits< _Alloc >::allocate [inline], [static]
```

Allocate memory.

**Parameters**

|                                      |                                              |
|--------------------------------------|----------------------------------------------|
| $\leftrightarrow$<br><code>_a</code> | An allocator.                                |
| $\leftrightarrow$<br><code>_n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

Definition at line 313 of file `bits/alloc_traits.h`.

**4.6.3.2 allocate()** [2/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator_traits< _Alloc >::allocate [inline], [static]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 328 of file `bits/alloc_traits.h`.

**4.6.3.3 allocate()** [3/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n) [inline], [static], [inherited]
```

Allocate memory.

**Parameters**

|                                      |                                              |
|--------------------------------------|----------------------------------------------|
| $\leftrightarrow$<br><code>_a</code> | An allocator.                                |
| $\leftrightarrow$<br><code>_n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

Definition at line 313 of file `bits/alloc_traits.h`.

Referenced by `std::__allocate_guarded()`.

#### 4.6.3.4 `allocate()` [4/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static], [inherited]
```

Allocate memory.

##### Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

##### Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 328 of file `bits/alloc_traits.h`.

#### 4.6.3.5 `construct()` [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp , typename... _Args>
static constexpr auto std::allocator_traits< _Alloc >::construct (
 typename _Tp ,
 typename... _Args) [inline], [static], [noexcept]
```

Construct an object of type `_Tp`.

##### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args) ...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 356 of file `bits/alloc_traits.h`.

#### 4.6.3.6 `construct()` [2/2]

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
static constexpr auto std::allocator_traits< _Alloc >::construct (
 _Alloc & __a,
 _Tp * __p,
 _Args &&... __args) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__↵
args)...)) [inline], [static], [noexcept], [inherited]
```

Construct an object of type `_Tp`.

##### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args) ...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type `_Tp` at location `__p` from the arguments `__args...`

Definition at line 356 of file `bits/alloc_traits.h`.

#### 4.6.3.7 `deallocate()` [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr void std::allocator_traits< _Alloc >::deallocate [inline], [static]
```

Deallocate memory.

##### Parameters

|                        |                                                |
|------------------------|------------------------------------------------|
| <code>↵<br/>__a</code> | An allocator.                                  |
| <code>↵<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>↵<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

Definition at line 340 of file `bits/alloc_traits.h`.

#### 4.6.3.8 `deallocate()` [2/2]

```
template<typename _Alloc>
static constexpr void std::allocator_traits<_Alloc >::deallocate (
 _Alloc & __a,
 pointer __p,
 size_type __n) [inline], [static], [inherited]
```

Deallocate memory.

##### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

Definition at line 340 of file `bits/alloc_traits.h`.

Referenced by `std::__allocated_ptr<_Alloc >::~~__allocated_ptr()`.

#### 4.6.3.9 `destroy()` [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp >
static constexpr void std::allocator_traits<_Alloc >::destroy (
 typename _Tp) [inline], [static], [noexcept]
```

Destroy an object of type `_Tp`.

##### Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__a</code> | An allocator.                    |
| <code>__p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 372 of file `bits/alloc_traits.h`.

**4.6.3.10 destroy()** [2/2]

```
template<typename _Alloc>
template<typename _Tp >
static constexpr void std::allocator_traits< _Alloc >::destroy (
 _Alloc & __a,
 _Tp * __p) [inline], [static], [noexcept], [inherited]
```

Destroy an object of type `_Tp`.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__a</code> | An allocator.                    |
| <code>__p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

Definition at line 372 of file `bits/alloc_traits.h`.

Referenced by `std::_Destroy()`.

**4.6.3.11 max\_size()** [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr size_type std::allocator_traits< _Alloc >::max_size [inline], [static], [noexcept]
```

The maximum supported allocation size.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

**Returns**

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 385 of file `bits/alloc_traits.h`.



4.6.3.12 `max_size()` [2/2]

```
template<typename _Alloc>
static constexpr size_type std::allocator_traits<_Alloc>::max_size (
 const _Alloc & __a) [inline], [static], [noexcept], [inherited]
```

The maximum supported allocation size.

## Parameters

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

## Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 385 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`, and `std::list<__inp, __rebind_inp>::max_size()`.

4.6.3.13 `select_on_container_copy_construction()`

```
template<typename _Alloc>
static constexpr _Alloc std::allocator_traits<_Alloc>::select_on_container_copy_construction (
 const _Alloc & __rhs) [inline], [static], [inherited]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 397 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [ext/alloc\\_traits.h](#)

## 4.7 std::\_\_allocated\_ptr< \_Alloc > Struct Template Reference

### Public Types

- using **pointer** = typename `allocator_traits< _Alloc >::pointer`
- using **value\_type** = typename `allocator_traits< _Alloc >::value_type`

### Public Member Functions

- `__allocated_ptr ( _Alloc &__a, pointer __ptr ) noexcept`
- `template<typename _Ptr , typename _Req = _Require<is_same<_Ptr, value_type*>>>> __allocated_ptr ( _Alloc &__a, _Ptr __ptr )`
- `__allocated_ptr ( __allocated_ptr &&__gd ) noexcept`
- `~__allocated_ptr ()`
- `value_type * get ()`
- `__allocated_ptr & operator= (std::nullptr_t) noexcept`

#### 4.7.1 Detailed Description

```
template<typename _Alloc>
struct std::__allocated_ptr< _Alloc >
```

Non-standard RAII type for managing pointers obtained from allocators.

Definition at line 46 of file `allocated_ptr.h`.

#### 4.7.2 Constructor & Destructor Documentation

##### 4.7.2.1 \_\_allocated\_ptr() [1/3]

```
template<typename _Alloc >
std::__allocated_ptr< _Alloc >::__allocated_ptr (
 _Alloc & __a,
 pointer __ptr) [inline], [noexcept]
```

Take ownership of `__ptr`.

Definition at line 52 of file `allocated_ptr.h`.

## 4.7.2.2 \_\_allocated\_ptr() [2/3]

```
template<typename _Alloc >
template<typename _Ptr , typename _Req = _Require<is_same<_Ptr, value_type*>>>
std::__allocated_ptr< _Alloc >::__allocated_ptr (
 _Alloc & __a,
 _Ptr __ptr) [inline]
```

Convert \_\_ptr to allocator's pointer type and take ownership of it.

Definition at line 59 of file allocated\_ptr.h.

## 4.7.2.3 \_\_allocated\_ptr() [3/3]

```
template<typename _Alloc >
std::__allocated_ptr< _Alloc >::__allocated_ptr (
 __allocated_ptr< _Alloc > && __gd) [inline], [noexcept]
```

Transfer ownership of the owned pointer.

Definition at line 65 of file allocated\_ptr.h.

## 4.7.2.4 ~\_\_allocated\_ptr()

```
template<typename _Alloc >
std::__allocated_ptr< _Alloc >::~~__allocated_ptr () [inline]
```

Deallocate the owned pointer.

Definition at line 70 of file allocated\_ptr.h.

References std::allocator\_traits<\_Alloc>::deallocate().

## 4.7.3 Member Function Documentation

## 4.7.3.1 get()

```
template<typename _Alloc >
value_type* std::__allocated_ptr< _Alloc >::get (
 void) [inline]
```

Get the address that the owned pointer refers to.

Definition at line 85 of file allocated\_ptr.h.

#### 4.7.3.2 operator=()

```
template<typename _Alloc >
__allocated_ptr& std::__allocated_ptr< _Alloc >::operator= (
 std::nullptr_t) [inline], [noexcept]
```

Release ownership of the owned pointer.

Definition at line 78 of file `allocated_ptr.h`.

The documentation for this struct was generated from the following file:

- [allocated\\_ptr.h](#)

### 4.8 std::\_\_atomic\_base< \_ITp > Struct Template Reference

#### Public Types

- using **difference\_type** = value\_type
- using **value\_type** = \_ITp

#### Public Member Functions

- **\_\_atomic\_base** (const [\\_\\_atomic\\_base](#) &)=delete
- constexpr **\_\_atomic\_base** (\_\_int\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

- `__int_type fetch_sub (__int_type __i, memory\_order __m=memory_order_seq_cst)` noexcept
- `__int_type fetch_sub (__int_type __i, memory\_order __m=memory_order_seq_cst)` volatile noexcept
- `__int_type fetch_xor (__int_type __i, memory\_order __m=memory_order_seq_cst)` noexcept
- `__int_type fetch_xor (__int_type __i, memory\_order __m=memory_order_seq_cst)` volatile noexcept
- `bool is_lock_free ()` const noexcept
- `bool is_lock_free ()` const volatile noexcept
- `__int_type load (memory\_order __m=memory_order_seq_cst)` const noexcept
- `__int_type load (memory\_order __m=memory_order_seq_cst)` const volatile noexcept
- `operator __int_type ()` const noexcept
- `operator __int_type ()` const volatile noexcept
- `__int_type operator&= (__int_type __i)` noexcept
- `__int_type operator&= (__int_type __i)` volatile noexcept
- `__int_type operator++ (int)` noexcept
- `__int_type operator++ (int)` volatile noexcept
- `__int_type operator++ ()` noexcept
- `__int_type operator++ ()` volatile noexcept
- `__int_type operator+= (__int_type __i)` noexcept
- `__int_type operator+= (__int_type __i)` volatile noexcept
- `__int_type operator-- (int)` noexcept
- `__int_type operator-- (int)` volatile noexcept
- `__int_type operator-- ()` noexcept
- `__int_type operator-- ()` volatile noexcept
- `__int_type operator-= (__int_type __i)` noexcept
- `__int_type operator-= (__int_type __i)` volatile noexcept
- `\_\_atomic\_base & operator= (const \_\_atomic\_base &)=delete`
- `\_\_atomic\_base & operator= (const \_\_atomic\_base &) volatile=delete`
- `__int_type operator= (__int_type __i)` noexcept
- `__int_type operator= (__int_type __i)` volatile noexcept
- `__int_type operator^= (__int_type __i)` noexcept
- `__int_type operator^= (__int_type __i)` volatile noexcept
- `__int_type operator|= (__int_type __i)` noexcept
- `__int_type operator|= (__int_type __i)` volatile noexcept
- `void store (__int_type __i, memory\_order __m=memory_order_seq_cst)` noexcept
- `void store (__int_type __i, memory\_order __m=memory_order_seq_cst)` volatile noexcept

#### 4.8.1 Detailed Description

```
template<typename _ITp>
struct std::atomic_base<_ITp>
```

Base class for atomic integrals.

Definition at line 140 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.9 std::\_\_atomic\_base<\_PTp\*> Struct Template Reference

### Public Member Functions

- **\_\_atomic\_base** (const **\_\_atomic\_base** &)=delete
- constexpr **\_\_atomic\_base** (\_\_pointer\_type \_\_p) noexcept
- bool **compare\_exchange\_strong** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, **memory\_order** \_\_m1, **memory\_order** \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_pointer\_type &\_\_p1, \_\_pointer\_type \_\_p2, **memory\_order** \_\_m1, **memory\_order** \_\_m2) volatile noexcept
- \_\_pointer\_type **exchange** (\_\_pointer\_type \_\_p, **memory\_order** \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **exchange** (\_\_pointer\_type \_\_p, **memory\_order** \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_pointer\_type **fetch\_add** (ptrdiff\_t \_\_d, **memory\_order** \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **fetch\_add** (ptrdiff\_t \_\_d, **memory\_order** \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_pointer\_type **fetch\_sub** (ptrdiff\_t \_\_d, **memory\_order** \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_pointer\_type **fetch\_sub** (ptrdiff\_t \_\_d, **memory\_order** \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_pointer\_type **load** (**memory\_order** \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_pointer\_type **load** (**memory\_order** \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_pointer\_type** () const noexcept
- **operator \_\_pointer\_type** () const volatile noexcept
- \_\_pointer\_type **operator++** (int) noexcept
- \_\_pointer\_type **operator++** (int) volatile noexcept
- \_\_pointer\_type **operator++** () noexcept
- \_\_pointer\_type **operator++** () volatile noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator+=** (ptrdiff\_t \_\_d) volatile noexcept
- \_\_pointer\_type **operator--** (int) noexcept
- \_\_pointer\_type **operator--** (int) volatile noexcept
- \_\_pointer\_type **operator--** () noexcept
- \_\_pointer\_type **operator--** () volatile noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) noexcept
- \_\_pointer\_type **operator-=** (ptrdiff\_t \_\_d) volatile noexcept
- **\_\_atomic\_base** & **operator=** (const **\_\_atomic\_base** &)=delete
- **\_\_atomic\_base** & **operator=** (const **\_\_atomic\_base** &) volatile=delete
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) noexcept
- \_\_pointer\_type **operator=** (\_\_pointer\_type \_\_p) volatile noexcept
- void **store** (\_\_pointer\_type \_\_p, **memory\_order** \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (\_\_pointer\_type \_\_p, **memory\_order** \_\_m=memory\_order\_seq\_cst) volatile noexcept

### 4.9.1 Detailed Description

```
template<typename _PTp>
struct std::__atomic_base<_PTp*>
```

Partial specialization for pointer types.

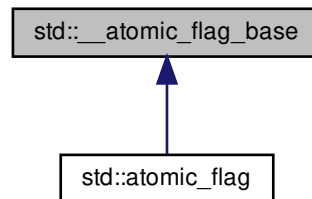
Definition at line 599 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

## 4.10 std::\_\_atomic\_flag\_base Struct Reference

Inheritance diagram for std::\_\_atomic\_flag\_base:



### Public Attributes

- \_\_atomic\_flag\_data\_type **M**i

#### 4.10.1 Detailed Description

Base type for atomic\_flag.

Base type is POD with data, allowing atomic\_flag to derive from it and meet the standard layout type requirement. In addition to compatibility with a C interface, this allows different implementations of atomic\_flag to use the same atomic operation functions, via a standard conversion to the \_\_atomic\_flag\_base argument.

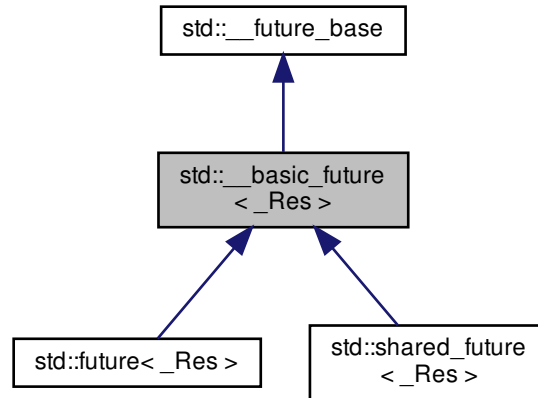
Definition at line 176 of file atomic\_base.h.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

#### 4.11 `std::__basic_future<_Res>` Class Template Reference

Inheritance diagram for `std::__basic_future<_Res>`:



##### Public Types

- `template<typename _Res>`  
`using _Ptr = unique\_ptr<_Res, \_Result\_base::Deleter>`
- `using \_State\_base = \_State\_baseV2`

##### Public Member Functions

- `\_\_basic\_future (const \_\_basic\_future &)=delete`
- `\_\_basic\_future & operator= (const \_\_basic\_future &)=delete`
- `bool valid () const noexcept`
- `void wait () const`
- `template<typename _Rep, typename _Period>`  
`future\_status wait\_for (const chrono::duration<_Rep, _Period> &__rel) const`
- `template<typename _Clock, typename _Duration>`  
`future\_status wait\_until (const chrono::time\_point<_Clock, _Duration> &__abs) const`

##### Static Public Member Functions

- `template<typename _Res, typename _Allocator>`  
`static \_Ptr<\_Result\_alloc<_Res, _Allocator>> \_S\_allocate\_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp>`  
`static \_Ptr<\_Result<_Res>> \_S\_allocate\_result (const std::allocator<_Tp> &__a)`
- `template<typename _BoundFn>`  
`static std::shared\_ptr<_State_base> \_S\_make\_async\_state (_BoundFn &&__fn)`
- `template<typename _BoundFn>`  
`static std::shared\_ptr<_State_base> \_S\_make\_deferred\_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn>`  
`static \_Task\_setter<_Res_ptr, _BoundFn> \_S\_task\_setter (_Res_ptr &__ptr, _BoundFn &__call)`



### Protected Types

- typedef `__future_base::Result<_Res>` & `__result_type`
- typedef `shared_ptr<_State_base>` `__state_type`

### Protected Member Functions

- `__basic_future` (const `__state_type` & `__state`)
- `__basic_future` (const `shared_future<_Res>` &) noexcept
- `__basic_future` (`shared_future<_Res>` &&) noexcept
- `__basic_future` (`future<_Res>` &&) noexcept
- `__result_type _M_get_result` () const
- void `_M_swap` (`__basic_future` & `__that`) noexcept

#### 4.11.1 Detailed Description

```
template<typename _Res>
class std::__basic_future<_Res>
```

Common implementation for future and shared\_future.

Definition at line 682 of file future.

#### 4.11.2 Member Typedef Documentation

##### 4.11.2.1 `_Ptr`

```
template<typename _Res>
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file future.

#### 4.11.3 Member Function Documentation

#### 4.11.3.1 `_M_get_result()`

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

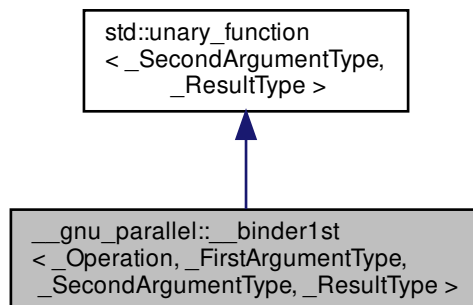
Definition at line 725 of file future.

The documentation for this class was generated from the following file:

- [future](#)

#### 4.12 `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



#### Public Types

- typedef `_SecondArgumentType` [argument\\_type](#)
- typedef `_ResultType` [result\\_type](#)

#### Public Member Functions

- `__binder1st` (const `_Operation` &\_\_x, const `_FirstArgumentType` &\_\_y)
- `_ResultType operator()` (const `_SecondArgumentType` &\_\_x)
- `_ResultType operator()` (`_SecondArgumentType` &\_\_x) const

#### Protected Attributes

- `_Operation` **`_M_op`**
- `_FirstArgumentType` **`_M_value`**

#### 4.12.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder1st<_Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder1st`, but giving the argument types explicitly.

Definition at line 192 of file `base.h`.

#### 4.12.2 Member Typedef Documentation

##### 4.12.2.1 `argument_type`

```
typedef _SecondArgumentType std::unary_function<_SecondArgumentType, _ResultType >::argument_type
[inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 4.12.2.2 `result_type`

```
typedef _ResultType std::unary_function<_SecondArgumentType, _ResultType >::result_type [inherited]
```

`result_type` is the return type

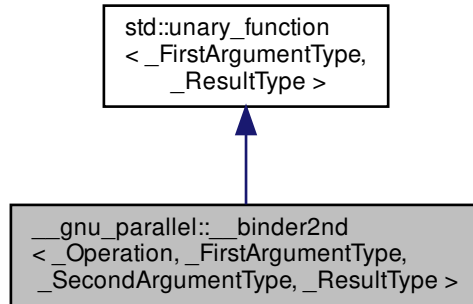
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [base.h](#)

#### 4.13 `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >` Class Template Reference

Inheritance diagram for `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`:



##### Public Types

- typedef `_FirstArgumentType` `argument_type`
- typedef `_ResultType` `result_type`

##### Public Member Functions

- `__binder2nd` (`const _Operation &__x, const _SecondArgumentType &__y`)
- `_ResultType operator()` (`const _FirstArgumentType &__x`) `const`
- `_ResultType operator()` (`_FirstArgumentType &__x`)

##### Protected Attributes

- `_Operation` `_M_op`
- `_SecondArgumentType` `_M_value`

##### 4.13.1 Detailed Description

```
template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
```

Similar to `std::binder2nd`, but giving the argument types explicitly.

Definition at line 220 of file `base.h`.

## 4.13.2 Member Typedef Documentation

## 4.13.2.1 argument\_type

```
typedef _FirstArgumentType std::unary_function< _FirstArgumentType , _ResultType >::argument_type
[inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 4.13.2.2 result\_type

```
typedef _ResultType std::unary_function< _FirstArgumentType , _ResultType >::result_type [inherited]
```

result\_type is the return type

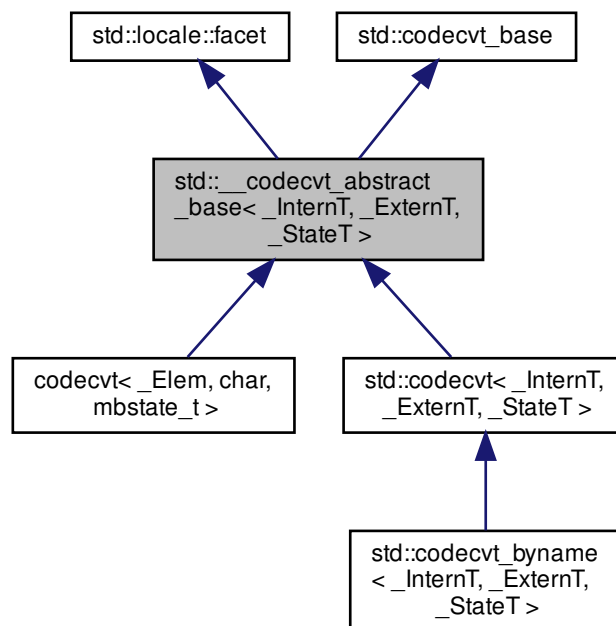
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [base.h](#)

## 4.14 std::\_\_codecvt\_abstract\_base&lt;\_InternT, \_ExternT, \_StateT&gt; Class Template Reference

Inheritance diagram for std::\_\_codecvt\_abstract\_base<\_InternT, \_ExternT, \_StateT>:



## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Protected Member Functions

- `__codecvt_abstract_base` (size\_t \_\_refs=0)
- virtual bool **do\_always\_noconv** () const =0 throw ()
- virtual int **do\_encoding** () const =0 throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const =0
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const =0
- virtual int **do\_max\_length** () const =0 throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const =0

## Static Protected Member Functions

- static `__c_locale` **\_S\_clone\_c\_locale** (`__c_locale` &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (`__c_locale` &\_\_cloc, const char \*\_\_s, `__c_locale` \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (`__c_locale` &\_\_cloc)
- static `__c_locale` **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static `__c_locale` **\_S\_lc\_ctype\_c\_locale** (`__c_locale` \_\_cloc, const char \*\_\_s)

## 4.14.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>
```

Common base for codecvt functions.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 68 of file `codecvt.h`.

## 4.14.2 Member Function Documentation

4.14.2.1 `do_out()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT>::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [pure virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

## See also

`out` for more information.

Implemented in `std::codecvt<char32_t, char, mbstate_t>`, `std::codecvt<char16_t, char, mbstate_t>`, `std::codecvt<wchar_t, char, mbstate_t>`, `std::codecvt<char, char, mbstate_t>`, `std::codecvt<_InternT, _ExternT, _StateT>`, `std::codecvt<_Elem, char, mbstate_t>`, and `std::codecvt<_InternT, _ExternT, encoding_state>`.

Referenced by `std::__codecvt_abstract_base<char32_t, char, mbstate_t>::out()`.

#### 4.14.2.2 in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

##### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

##### Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.



4.14.2.3 `out()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base<_InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

#### 4.14.2.4 unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

##### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

##### Returns

`codecvt_base::result`.

Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.15 \_\_gnu\_cxx::\_\_common\_pool\_policy< \_PoolTp, \_Thread > Struct Template Reference

Inherits `__gnu_cxx::__common_pool_base< _PoolTp, _Thread >`.

## 4.15.1 Detailed Description

```
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
```

Policy for shared \_\_pool objects.

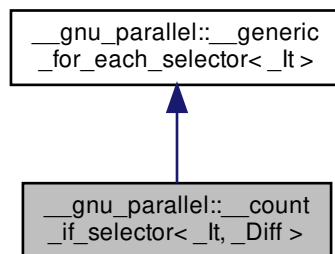
Definition at line 459 of file mt\_allocator.h.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 4.16 \_\_gnu\_parallel::\_\_count\_if\_selector&lt; \_It, \_Diff &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_count\_if\_selector< \_It, \_Diff >:



## Public Member Functions

- `template<typename _Op >`  
`_Diff operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.16.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector< _It, _Diff >
```

std::count\_if () selector.

Definition at line 194 of file for\_each\_selectors.h.

## 4.16.2 Member Function Documentation

### 4.16.2.1 operator()()

```
template<typename _It, typename _Diff>
template<typename _Op >
_Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                                  |                              |
|----------------------------------|------------------------------|
| <a href="#"><code>__o</code></a> | Operator.                    |
| <a href="#"><code>__i</code></a> | iterator referencing object. |

#### Returns

1 if count, 0 if does not count.

Definition at line 202 of file `for_each_selectors.h`.

## 4.16.3 Member Data Documentation

### 4.16.3.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

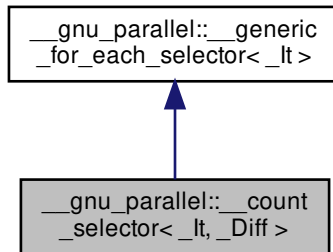
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.17 \_\_gnu\_parallel::\_\_count\_selector&lt;\_It, \_Diff&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_count\_selector<\_It, \_Diff>:



## Public Member Functions

- `template<typename _ValueType>`  
`_Diff operator() (_ValueType &__v, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.17.1 Detailed Description

```
template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector<_It, _Diff>
```

`std::count()` selector.

Definition at line 180 of file `for_each_selectors.h`.

## 4.17.2 Member Function Documentation

## 4.17.2.1 operator&gt;()

```
template<typename _It, typename _Diff>
template<typename _ValueType>
_Diff __gnu_parallel::__count_selector<_It, _Diff>::operator() (
 _ValueType & __v,
 _It __i) [inline]
```

Functor execution.

**Parameters**

|                                        |                              |
|----------------------------------------|------------------------------|
| <a href="#"><code>_↵<br/>_v</code></a> | Current value.               |
| <a href="#"><code>_↵<br/>_i</code></a> | iterator referencing object. |

**Returns**

1 if count, 0 if does not count.

Definition at line 188 of file `for_each_selectors.h`.

**4.17.3 Member Data Documentation****4.17.3.1 `_M_finish_iterator`**

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

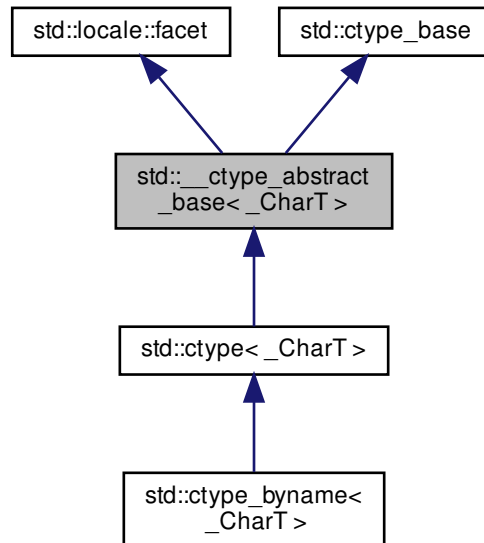
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.18 std::\_\_ctype\_abstract\_base&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::\_\_ctype\_abstract\_base<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef `_CharT` **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- bool **is** (mask `__m`, `char_type` `__c`) const
- const `char_type` \* **is** (const `char_type` \* `__lo`, const `char_type` \* `__hi`, mask \* `__vec`) const
- char **narrow** (`char_type` `__c`, char `__default`) const
- const `char_type` \* **narrow** (const `char_type` \* `__lo`, const `char_type` \* `__hi`, char `__default`, char \* `__to`) const
- const `char_type` \* **scan\_is** (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- const `char_type` \* **scan\_not** (mask `__m`, const `char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` **tolower** (`char_type` `__c`) const
- const `char_type` \* **tolower** (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` **toupper** (`char_type` `__c`) const
- const `char_type` \* **toupper** (`char_type` \* `__lo`, const `char_type` \* `__hi`) const
- `char_type` **widen** (char `__c`) const
- const char \* **widen** (const char \* `__lo`, const char \* `__hi`, `char_type` \* `__to`) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- **\_\_ctype\_abstract\_base** (size\_t \_\_refs=0)
- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const =0
- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const =0
- virtual char **do\_narrow** (char\_type \_\_c, char \_\_default) const =0
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const =0
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_tolower** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_toupper** (char\_type \_\_c) const =0
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const =0
- virtual char\_type **do\_widen** (char \_\_c) const =0
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const =0

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 4.18.1 Detailed Description

```
template<typename _CharT>
class std::__ctype_abstract_base< _CharT >
```

Common base for ctype facet.

This template class provides implementations of the public functions that forward to the protected virtual functions.

This template also provides abstract stubs for the protected virtual functions.

Definition at line 150 of file locale\_facets.h.



## 4.18.2 Member Typedef Documentation

## 4.18.2.1 char\_type

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base< _CharT >::char_type
```

Typedef for the template parameter.

Definition at line 155 of file locale\_facets.h.

## 4.18.3 Member Function Documentation

## 4.18.3.1 do\_is() [1/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
 mask __m,
 char_type __c) const [protected], [pure virtual]
```

Test char\_type classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| $\leftrightarrow$<br>__c | The char_type to find the mask of. |
| $\leftrightarrow$<br>__m | The mask to compare against.       |

## Returns

(*M* & \_\_m) != 0.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::is\(\)](#).

#### 4.18.3.2 `do_is()` [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [pure virtual]
```

Return a mask array.

This function finds the mask for each `char_type` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

##### Returns

`__hi`.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

#### 4.18.3.3 `do_narrow()` [1/2]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
 char_type __c,
 char __dfault) const [protected], [pure virtual]
```

Narrow `char_type` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

##### Parameters

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**

The converted char.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::narrow\(\)](#).

**4.18.3.4 do\_narrow()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [pure virtual]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [[\\_\\_lo](#),[\\_\\_hi](#)) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, [\\_\\_default](#) is used instead.

[do\\_narrow\(\)](#) is a hook for a derived facet to change the behavior of narrowing. [do\\_narrow\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

**Parameters**

|                           |                                   |
|---------------------------|-----------------------------------|
| <a href="#">__lo</a>      | Pointer to start of range.        |
| <a href="#">__hi</a>      | Pointer to end of range.          |
| <a href="#">__default</a> | Char to use if conversion fails.  |
| <a href="#">__to</a>      | Pointer to the destination array. |

**Returns**

[\\_\\_hi](#).

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

**4.18.3.5 do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_scan_is (
```

```

mask __m,
const char_type * __lo,
const char_type * __hi) const [protected], [pure virtual]

```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

#### Returns

Pointer to a matching char\_type if found, else `__hi`.

Implemented in `std::ctype< wchar_t >`, and `std::ctype< _CharT >`.

Referenced by `std::__ctype_abstract_base< wchar_t >::scan_is()`.

#### 4.18.3.6 do\_scan\_not()

```

template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]

```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [`lo`,`hi`) for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else \_\_hi.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::scan\\_not\(\)](#).

**4.18.3.7 do\_tolower()** [1/2]


```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_tolower (
 char_type __c) const [protected], [pure virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                     |                           |
|-------------------------------------------------------------------------------------|---------------------------|
|  | The char_type to convert. |
| __c                                                                                 |                           |

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Implemented in [std::ctype<wchar\\_t>](#), and [std::ctype<\\_CharT>](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base<wchar\\_t>::tolower\(\)](#).

**4.18.3.8 do\_tolower()** [2/2]



```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base<_CharT>::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [pure virtual]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                                       |                            |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>_lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>_hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

**4.18.3.9 do\_toupper()** [1/2]


```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
 char_type __c) const [protected], [pure virtual]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                         |                                        |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>char_type</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

Referenced by `std::__ctype_abstract_base< wchar_t >::toupper()`.

**4.18.3.10 do\_toupper()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype_abstract_base< _CharT >::do_toupper (
```



```
char_type * __lo,
const char_type * __hi) const [protected], [pure virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                                                                                                           |                            |
|-----------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br>__lo | Pointer to start of range. |
| <a href="#"></a><br>__hi | Pointer to end of range.   |

#### Returns

\_\_hi.

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

#### 4.18.3.11 do\_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base<_CharT>::do_widen (
 char __c) const [protected], [pure virtual]
```


Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                            |                      |
|------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br>__c | The char to convert. |
|------------------------------------------------------------------------------------------------------------|----------------------|

#### Returns

The converted char\_type

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

Referenced by [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >::widen\(\)](#).

#### 4.18.3.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::__ctype_abstract_base< _CharT >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [pure virtual]
```

Widen char array.

This function converts each char in the input to [char\\_type](#) using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

##### Parameters

|                            |                                   |
|----------------------------|-----------------------------------|
| <a href="#">_↵<br/>_lo</a> | Pointer to start range.           |
| <a href="#">_↵<br/>_hi</a> | Pointer to end of range.          |
| <a href="#">_↵<br/>_to</a> | Pointer to the destination array. |

##### Returns

[\\_\\_hi](#).

Implemented in [std::ctype< wchar\\_t >](#), and [std::ctype< \\_CharT >](#).

#### 4.18.3.13 is() [1/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
 mask __m,
 char_type __c) const [inline]
```

Test [char\\_type](#) classification.

This function finds a mask M for [\\_\\_c](#) and compares it to mask [\\_\\_m](#). It does so by returning the value of [ctype<char\\_↵  
type>::do\\_is\(\)](#).



## Parameters

|                        |                                       |
|------------------------|---------------------------------------|
| $\hookleftarrow$<br>_c | The char_type to compare the mask of. |
| $\hookleftarrow$<br>_m | The mask to compare against.          |

## Returns

(M & \_\_m) != 0.

Definition at line 169 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter>::get().

## 4.18.3.14 is() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

## Parameters

|       |                                      |
|-------|--------------------------------------|
| __lo  | Pointer to start of range.           |
| __hi  | Pointer to end of range.             |
| __vec | Pointer to an array of mask storage. |

## Returns

\_\_hi.

Definition at line 186 of file locale\_facets.h.

## 4.18.3.15 narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
```

```
char_type __c,
char __default) const [inline]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Definition at line 331 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter >::get(), and std::time\_put<\_CharT, \_OutIter >::put().

#### 4.18.3.16 narrow() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, default is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**`__hi`.

Definition at line 353 of file locale\_facets.h.

**4.18.3.17 scan\_is()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against. |
| <code>↔<br/>__lo</code> | Pointer to start of range.   |
| <code>↔<br/>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to matching char\_type if found, else `__hi`.

Definition at line 202 of file locale\_facets.h.




**4.18.3.18 scan\_not()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                          |                                 |
|------------------------------------------------------------------------------------------|---------------------------------|
| <br>_m  | The mask to compare against.    |
| <br>_lo | Pointer to first char in range. |
| <br>_hi | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else \_\_hi.

Definition at line 218 of file locale\_facets.h.


**4.18.3.19 tolower()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

**Parameters**

|                                                                                           |                           |
|-------------------------------------------------------------------------------------------|---------------------------|
| <br>_c | The char_type to convert. |
|-------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else \_\_c.

Definition at line 261 of file locale\_facets.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::get().

**4.18.3.20 tolower()** [2/2]



```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::tolower (
```

```
char_type * __lo,
const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

## Parameters

|                                                                                           |                            |
|-------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

## Returns

\_\_hi.

Definition at line 276 of file locale\_facets.h.


## 4.18.3.21 toupper() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

## Parameters

|                                                                                            |                           |
|--------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|--------------------------------------------------------------------------------------------|---------------------------|

## Returns

The uppercase char\_type if convertible, else \_\_c.

Definition at line 232 of file locale\_facets.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::get().



## 4.18.3.22 toupper() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

## Parameters

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Definition at line 247 of file locale\_facets.h.

## 4.18.3.23 widen() [1/2]


```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
 char __c) const [inline]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                                                                                                                         |                      |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__c</code> | The char to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|

## Returns

The converted char\_type.

Definition at line 293 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter>::do\_get(), std::money\_get<\_CharT, \_InIter>::do\_get(), std::time\_put<\_CharT, \_OutIter>::do\_put(), and std::money\_put<\_CharT, \_OutIter>::do\_put().

## 4.18.3.24 widen() [2/2]

```
template<typename _CharT>
const char* std::__ctype_abstract_base<_CharT>::widen (
```

```
const char * __lo,
const char * __hi,
char_type * __to) const [inline]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                   |                                   |
|-----------------------------------|-----------------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start of range.        |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"><code>__to</code></a> | Pointer to the destination array. |

#### Returns

[`\_\_hi`](#).

Definition at line 312 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.19 std::\_\_detector< \_Default, \_AlwaysVoid, \_Op, \_Args > Struct Template Reference

### Public Types

- using **type** = \_Default
- using **value\_t** = [false\\_type](#)

#### 4.19.1 Detailed Description

```
template<typename _Default, typename _AlwaysVoid, template< typename... > class _Op, typename... _Args>
struct std::__detector< _Default, _AlwaysVoid, _Op, _Args >
```

Implementation of the detection idiom (negative case).

Definition at line 2582 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



4.20 `std::__detector<_Default, __void_t<_Op<_Args...>>, _Op, _Args...>` Struct Template Reference

## Public Types

- using **type** = `_Op<_Args...>`
- using **value\_t** = `true_type`

## 4.20.1 Detailed Description

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
struct std::__detector<_Default, __void_t<_Op<_Args...>>, _Op, _Args...>
```

Implementation of the detection idiom (positive case).

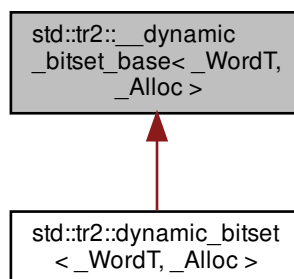
Definition at line 2591 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

4.21 `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>` Struct Template Reference

Inheritance diagram for `std::tr2::__dynamic_bitset_base<_WordT, _Alloc>`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_WordT` **block\_type**
- typedef `size_t` **size\_type**

## Public Member Functions

- `__dynamic_bitset_base` (const allocator\_type &\_\_alloc)
- `__dynamic_bitset_base` (const `__dynamic_bitset_base` &)=default
- `__dynamic_bitset_base` (`__dynamic_bitset_base` &&\_\_b)=default
- `__dynamic_bitset_base` (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type())
- `size_t _M_are_all_aux` () const noexcept
- `void _M_clear` () noexcept
- `void _M_do_and` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `void _M_do_append_block` (block\_type \_\_block, size\_type \_\_pos)
- `size_t _M_do_count` () const noexcept
- `void _M_do_dif` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `size_type _M_do_find_first` (size\_t \_\_not\_found) const
- `size_type _M_do_find_next` (size\_t \_\_prev, size\_t \_\_not\_found) const
- `void _M_do_flip` () noexcept
- `void _M_do_left_shift` (size\_t \_\_shift)
- `void _M_do_or` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `void _M_do_reset` () noexcept
- `void _M_do_right_shift` (size\_t \_\_shift)
- `void _M_do_set` () noexcept
- `unsigned long long _M_do_to_ullong` () const
- `unsigned long _M_do_to_ulong` () const
- `void _M_do_xor` (const `__dynamic_bitset_base` &\_\_x) noexcept
- `allocator_type _M_get_allocator` () const noexcept
- `block_type & _M_getword` (size\_type \_\_pos) noexcept
- `block_type _M_getword` (size\_type \_\_pos) const noexcept
- `block_type & _M_hiword` () noexcept
- `block_type _M_hiword` () const noexcept
- `bool _M_is_any` () const noexcept
- `bool _M_is_equal` (const `__dynamic_bitset_base` &\_\_x) const noexcept
- `bool _M_is_less` (const `__dynamic_bitset_base` &\_\_x) const noexcept
- `bool _M_is_proper_subset_of` (const `__dynamic_bitset_base` &\_\_b) const noexcept
- `bool _M_is_subset_of` (const `__dynamic_bitset_base` &\_\_b) const noexcept
- `void _M_resize` (size\_t \_\_nbits, bool \_\_value)
- `size_type _M_size` () const noexcept
- `void _M_swap` (`__dynamic_bitset_base` &\_\_b) noexcept
- `__dynamic_bitset_base & operator=` (const `__dynamic_bitset_base` &)=default
- `__dynamic_bitset_base & operator=` (`__dynamic_bitset_base` &&)=default

## Static Public Member Functions

- `static block_type _S_maskbit` (size\_type \_\_pos) noexcept
- `static size_type _S_whichbit` (size\_type \_\_pos) noexcept
- `static size_type _S_whichbyte` (size\_type \_\_pos) noexcept
- `static size_type _S_whichword` (size\_type \_\_pos) noexcept

## Public Attributes

- `std::vector< block_type, allocator_type > _M_w`

## Static Public Attributes

- static const size\_type `_S_bits_per_block`
- static const size\_type `npos`

## 4.21.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base<_WordT, _Alloc>
```

Base class, general case.

See documentation for `dynamic_bitset`.

Definition at line 62 of file `dynamic_bitset`.

## 4.21.2 Member Data Documentation

4.21.2.1 `_M_w`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base<_WordT, _Alloc>::_M_w
```

0 is the least significant word.

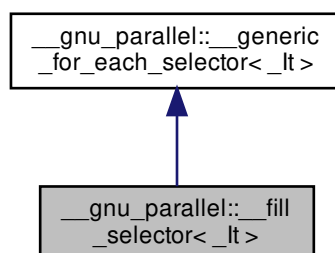
Definition at line 75 of file `dynamic_bitset`.

The documentation for this struct was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

4.22 `__gnu_parallel::__fill_selector<_It>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__fill_selector<_It>`:



## Public Member Functions

- `template<typename _ValueType >  
bool operator() (_ValueType &__v, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

### 4.22.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__fill_selector< _It >
```

`std::fill()` selector.

Definition at line 84 of file `for_each_selectors.h`.

### 4.22.2 Member Function Documentation

#### 4.22.2.1 operator>()

```
template<typename _It >
template<typename _ValueType >
bool __gnu_parallel::__fill_selector< _It >::operator() (
 _ValueType & __v,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                       |                              |
|-----------------------|------------------------------|
| <code>↔<br/>_v</code> | Current value.               |
| <code>↔<br/>_i</code> | iterator referencing object. |

Definition at line 91 of file `for_each_selectors.h`.

### 4.22.3 Member Data Documentation

4.22.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

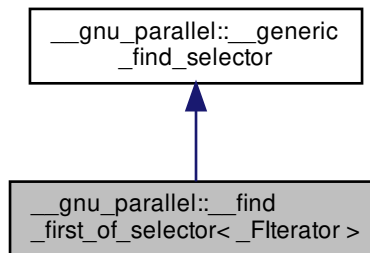
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

4.23 `__gnu_parallel::__find_first_of_selector<_FIterator>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__find_first_of_selector<_FIterator>`:



## Public Member Functions

- `__find_first_of_selector` (`_FIterator __begin`, `_FIterator __end`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair<_RAIter1, _RAIter2 > _M_sequential_algorithm` (`_RAIter1 __begin1`, `_RAIter1 __end1`, `_RAIter2 __begin2`, `_Pred __pred`)
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator()` (`_RAIter1 __i1`, `_RAIter2 __i2`, `_Pred __pred`)

## Public Attributes

- `_FIterator` `_M_begin`
- `_FIterator` `_M_end`

#### 4.23.1 Detailed Description

```
template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector< _FIterator >
```

Test predicate on several elements.

Definition at line 153 of file find\_selectors.h.

#### 4.23.2 Member Function Documentation

##### 4.23.2.1 \_M\_sequential\_algorithm()

```
template<typename _FIterator >
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_first_of_selector< _FIterator >::_M↵
sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

##### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

Definition at line 186 of file find\_selectors.h.

##### 4.23.2.2 operator()()

```
template<typename _FIterator >
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

## Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

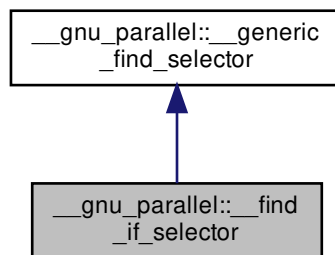
Definition at line 169 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 4.24 \_\_gnu\_parallel::\_\_find\_if\_selector Struct Reference

Inheritance diagram for `__gnu_parallel::__find_if_selector`:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair< _RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 4.24.1 Detailed Description

Test predicate on a single element, used for `std::find()` and `std::find_if()`.

Definition at line 50 of file `find_selectors.h`.

## 4.24.2 Member Function Documentation

### 4.24.2.1 `_M_sequential_algorithm()`

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]
```

Corresponding sequential algorithm on a sequence.

#### Parameters

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

Definition at line 72 of file `find_selectors.h`.

### 4.24.2.2 `operator()()`

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__find_if_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

Definition at line 60 of file `find_selectors.h`.

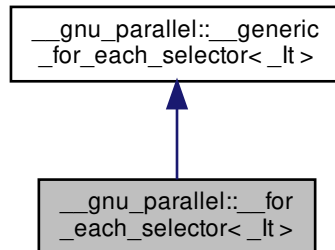
The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)



## 4.25 \_\_gnu\_parallel::\_\_for\_each\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_for\_each\_selector<\_It>:



## Public Member Functions

- `template<typename _Op>`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- `_It _M_finish_iterator`

## 4.25.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__for_each_selector<_It>
```

`std::for_each()` selector.

Definition at line 52 of file `for_each_selectors.h`.



## 4.25.2 Member Function Documentation

## 4.25.2.1 operator&gt;()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

**Parameters**

|                                                                                                                      |                              |
|----------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>_o</code> | Operator.                    |
| <a href="#"></a><br><code>_i</code> | iterator referencing object. |

Definition at line 59 of file `for_each_selectors.h`.

**4.25.3 Member Data Documentation****4.25.3.1 `_M_finish_iterator`**

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

**4.26 `__cxxabiv1::__forced_unwind` Class Reference****4.26.1 Detailed Description**

Thrown as part of forced unwinding.

A magic placeholder class that can be caught by reference to recognize forced unwinding.

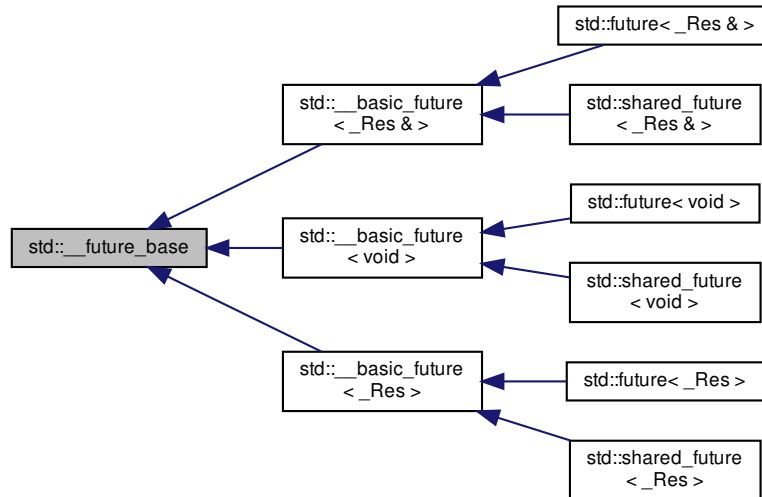
Definition at line 48 of file `cxxabi_forced.h`.

The documentation for this class was generated from the following file:

- [cxxabi\\_forced.h](#)

## 4.27 std::\_\_future\_base Struct Reference

Inheritance diagram for std::\_\_future\_base:



## Classes

- struct [\\_Result](#)
- struct [\\_Result<\\_Res &>](#)
- struct [\\_Result<void>](#)
- struct [\\_Result\\_alloc](#)
- struct [\\_Result\\_base](#)

## Public Types

- template<typename [\\_Res](#)>  
using [\\_Ptr](#) = [unique\\_ptr](#)<[\\_Res](#), [\\_Result\\_base::Deleter](#)>
- using [\\_State\\_base](#) = [\\_State\\_baseV2](#)

## Static Public Member Functions

- template<typename [\\_Res](#), typename [\\_Allocator](#)>  
static [\\_Ptr](#)<[\\_Result\\_alloc](#)<[\\_Res](#), [\\_Allocator](#)>> [\\_S\\_allocate\\_result](#) (const [\\_Allocator](#) &\_\_a)
- template<typename [\\_Res](#), typename [\\_Tp](#)>  
static [\\_Ptr](#)<[\\_Result](#)<[\\_Res](#)>> [\\_S\\_allocate\\_result](#) (const [std::allocator](#)<[\\_Tp](#)> &\_\_a)
- template<typename [\\_BoundFn](#)>  
static [std::shared\\_ptr](#)<[\\_State\\_base](#)> [\\_S\\_make\\_async\\_state](#) ([\\_BoundFn](#) &&\_\_fn)
- template<typename [\\_BoundFn](#)>  
static [std::shared\\_ptr](#)<[\\_State\\_base](#)> [\\_S\\_make\\_deferred\\_state](#) ([\\_BoundFn](#) &&\_\_fn)
- template<typename [\\_Res\\_ptr](#), typename [\\_BoundFn](#)>  
static [\\_Task\\_setter](#)<[\\_Res\\_ptr](#), [\\_BoundFn](#)> [\\_S\\_task\\_setter](#) ([\\_Res\\_ptr](#) &\_\_ptr, [\\_BoundFn](#) &\_\_call)

#### 4.27.1 Detailed Description

Base class and enclosing scope.

Definition at line 198 of file future.

#### 4.27.2 Member Typedef Documentation

##### 4.27.2.1 \_Ptr

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter>
```

A `unique_ptr` for result objects.

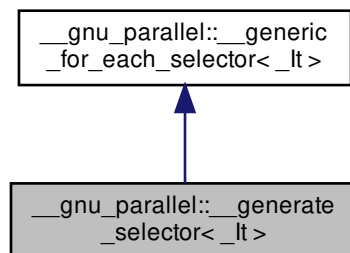
Definition at line 223 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

#### 4.28 \_\_gnu\_parallel::\_\_generate\_selector<\_It> Struct Template Reference

Inheritance diagram for `__gnu_parallel::__generate_selector<_It>`:



#### Public Member Functions

- `template<typename _Op >`  
`bool operator() (_Op &__o, _It __i)`

## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

## 4.28.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generate_selector<_It>
```

std::generate() selector.

Definition at line 68 of file `for_each_selectors.h`.

## 4.28.2 Member Function Documentation

## 4.28.2.1 operator&gt;()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

## Parameters

|                           |                              |
|---------------------------|------------------------------|
| <a href="#">_↔<br/>_o</a> | Operator.                    |
| <a href="#">_↔<br/>_i</a> | iterator referencing object. |

Definition at line 75 of file `for_each_selectors.h`.

## 4.28.3 Member Data Documentation

## 4.28.3.1 \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

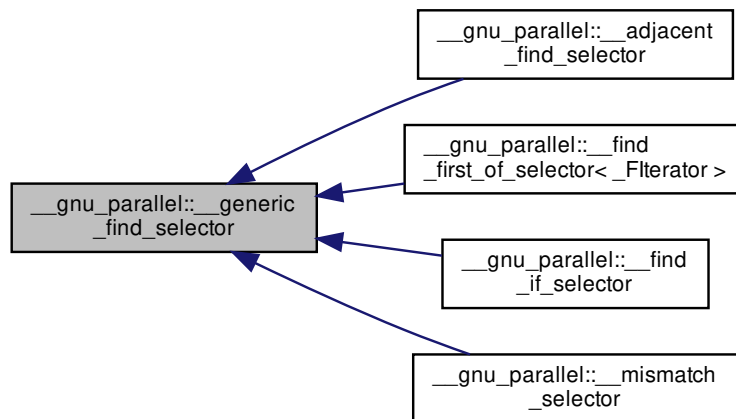
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.29 `__gnu_parallel::__generic_find_selector` Struct Reference

Inheritance diagram for `__gnu_parallel::__generic_find_selector`:



### 4.29.1 Detailed Description

Base class of all `__gnu_parallel::__find_template` selectors.

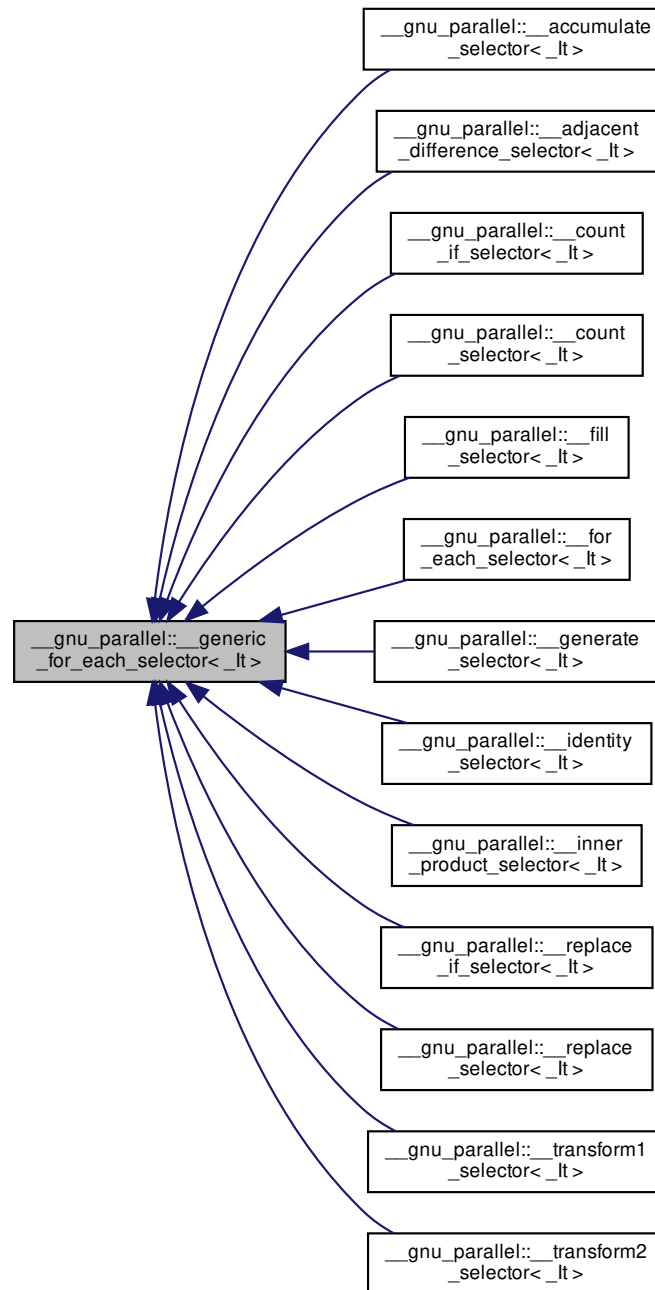
Definition at line 43 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

## 4.30 \_\_gnu\_parallel::\_\_generic\_for\_each\_selector&lt;\_It&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_generic\_for\_each\_selector<\_It>:



## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

## 4.30.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__generic_for_each_selector< _It >
```

Generic \_\_selector for embarrassingly parallel functions.

Definition at line 42 of file `for_each_selectors.h`.

## 4.30.2 Member Data Documentation

## 4.30.2.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

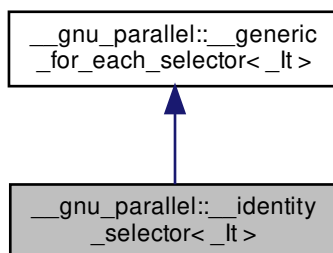
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.31 \_\_gnu\_parallel::\_\_identity\_selector&lt; \_It &gt; Struct Template Reference

Inheritance diagram for `__gnu_parallel::__identity_selector< _It >`:





### Public Member Functions

- `template<typename _Op>  
_It operator() (_Op __o, _It __i)`

### Public Attributes

- `_It _M_finish_iterator`

#### 4.31.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__identity_selector<_It>
```

Selector that just returns the passed iterator.

Definition at line 253 of file `for_each_selectors.h`.

#### 4.31.2 Member Function Documentation

##### 4.31.2.1 `operator()()`

```
template<typename _It>
template<typename _Op>
_It __gnu_parallel::__identity_selector<_It>::operator() (
 _Op __o,
 _It __i) [inline]
```

Functor execution.

#### Parameters

|                       |                              |
|-----------------------|------------------------------|
| <code>↵<br/>_o</code> | Operator (unused).           |
| <code>↵<br/>_i</code> | iterator referencing object. |

#### Returns

Passed iterator.

Definition at line 261 of file `for_each_selectors.h`.

### 4.31.3 Member Data Documentation

#### 4.31.3.1 \_M\_finish\_iterator

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

\_Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

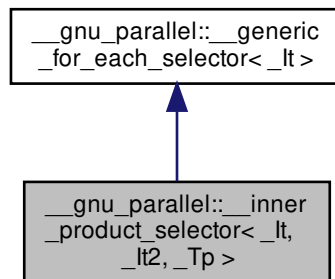
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.32 \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp > Struct Template Reference

Inheritance diagram for `__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >`:



#### Public Member Functions

- [\\_\\_inner\\_product\\_selector](#) (`_It __b1, _It2 __b2`)
- `template<typename _Op >`  
`_Tp operator()` (`_Op __mult, _It __current`)

#### Public Attributes

- `_It` [\\_\\_begin1\\_iterator](#)
- `_It2` [\\_\\_begin2\\_iterator](#)
- `_It` [\\_M\\_finish\\_iterator](#)

## 4.32.1 Detailed Description

```
template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>
```

`std::inner_product()` selector.

Definition at line 222 of file `for_each_selectors.h`.

## 4.32.2 Constructor &amp; Destructor Documentation

4.32.2.1 `__inner_product_selector()`

```
template<typename _It , typename _It2 , typename _Tp >
__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__inner_product_selector (
 _It __b1,
 _It2 __b2) [inline], [explicit]
```

Constructor.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__b1</code> | Begin iterator of first sequence.  |
| <code>__b2</code> | Begin iterator of second sequence. |

Definition at line 234 of file `for_each_selectors.h`.

## 4.32.3 Member Function Documentation

4.32.3.1 `operator>()()`

```
template<typename _It , typename _It2 , typename _Tp >
template<typename _Op >
_Tp __gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator() (
 _Op __mult,
 _It __current) [inline]
```

Functor execution.

## Parameters

|                        |                              |
|------------------------|------------------------------|
| <code>__mult</code>    | Multiplication functor.      |
| <code>__current</code> | iterator referencing object. |

## Returns

Inner product elemental `__result`.

Definition at line 243 of file `for_each_selectors.h`.

References `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin1_iterator`, and `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::__begin2_iterator`.

## 4.32.4 Member Data Documentation

### 4.32.4.1 `__begin1_iterator`

```
template<typename _It , typename _It2 , typename _Tp >
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.

Definition at line 225 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()()`.

### 4.32.4.2 `__begin2_iterator`

```
template<typename _It , typename _It2 , typename _Tp >
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.

Definition at line 228 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>::operator()()`.

### 4.32.4.3 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.33 `std::__is_location_invariant< _Tp >` Struct Template Reference

Inherits type `< _Tp >`.

#### 4.33.1 Detailed Description

```
template<typename _Tp>
struct std::__is_location_invariant< _Tp >
```

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Trivially copyable types are location-invariant and users can specialize this trait for other types.

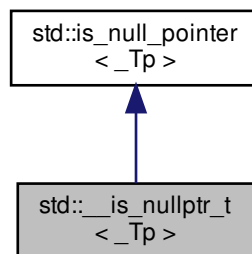
Definition at line 71 of file `std_function.h`.

The documentation for this struct was generated from the following file:

- [std\\_function.h](#)

### 4.34 `std::__is_nullptr_t< _Tp >` Struct Template Reference

Inheritance diagram for `std::__is_nullptr_t< _Tp >`:



#### 4.34.1 Detailed Description

```
template<typename _Tp>
struct std::__is_nullptr_t< _Tp >
```

`__is_nullptr_t` (deprecated extension).

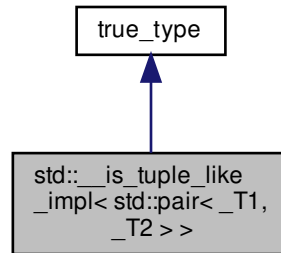
Definition at line 519 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 4.35 `std::__is_tuple_like_impl< std::pair< _T1, _T2 > >` Struct Template Reference

Inheritance diagram for `std::__is_tuple_like_impl< std::pair< _T1, _T2 > >`:



#### Public Types

- typedef [integral\\_constant< \\_Tp, \\_\\_v >](#) **type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 4.35.1 Detailed Description

```
template<typename _T1, typename _T2>
struct std::__is_tuple_like_impl< std::pair< _T1, _T2 > >
```

Partial specialization for `std::pair`.

Definition at line 152 of file `utility`.

The documentation for this struct was generated from the following file:

- [utility](#)

## 4.36 `__gnu_parallel::__max_element_reduct<_Compare, _It>` Struct Template Reference

### Public Member Functions

- `__max_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

#### 4.36.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.

Definition at line 321 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.37 `__gnu_parallel::__min_element_reduct<_Compare, _It>` Struct Template Reference

### Public Member Functions

- `__min_element_reduct` (`_Compare &__c`)
- `_It operator()` (`_It __x, _It __y`)

### Public Attributes

- `_Compare & __comp`

#### 4.37.1 Detailed Description

```
template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct<_Compare, _It>
```

Reduction for finding the maximum element, using a comparator.

Definition at line 307 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

### 4.38 `__gnu_cxx::__detail::__mini_vector<_Tp>` Class Template Reference

#### Public Types

- typedef const `_Tp` & **const\_reference**
- typedef `std::ptrdiff_t` **difference\_type**
- typedef pointer **iterator**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `std::size_t` **size\_type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator `__pos`) throw ()
- void **insert** (iterator `__pos`, const\_reference `__x`)
- reference **operator[]** (const size\_type `__pos`) const throw ()
- void **pop\_back** () throw ()
- void **push\_back** (const\_reference `__x`)
- size\_type **size** () const throw ()

#### 4.38.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::__mini_vector<_Tp>
```

`__mini_vector<>` is a stripped down version of the full-fledged `std::vector<>`.

It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present.
2. Used ONLY for PODs.
3. No Allocator template argument. Uses `operator new()` to get memory, and `operator delete()` to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

Definition at line 67 of file `bitmap_allocator.h`.

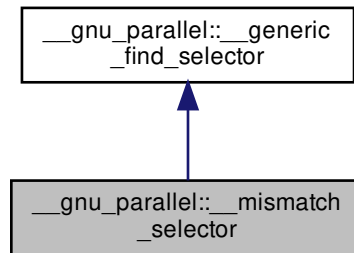
The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)



## 4.39 \_\_gnu\_parallel::\_\_mismatch\_selector Struct Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_mismatch\_selector:



## Public Member Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`std::pair<_RAIter1, _RAIter2 > \_M\_sequential\_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred >`  
`bool operator\(\) (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)`

## 4.39.1 Detailed Description

Test inverted predicate on a single element.

Definition at line 119 of file `find_selectors.h`.

## 4.39.2 Member Function Documentation

4.39.2.1 `_M_sequential_algorithm()`

```

template<typename _RAIter1, typename _RAIter2, typename _Pred >
std::pair<_RAIter1, _RAIter2> __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
 _RAIter1 __begin1,
 _RAIter1 __end1,
 _RAIter2 __begin2,
 _Pred __pred) [inline]

```

Corresponding sequential algorithm on a sequence.

**Parameters**

|                       |                                    |
|-----------------------|------------------------------------|
| <code>__begin1</code> | Begin iterator of first sequence.  |
| <code>__end1</code>   | End iterator of first sequence.    |
| <code>__begin2</code> | Begin iterator of second sequence. |
| <code>__pred</code>   | Find predicate.                    |

Definition at line 143 of file `find_selectors.h`.

**4.39.2.2 operator()()**

```
template<typename _RAIter1 , typename _RAIter2 , typename _Pred >
bool __gnu_parallel::__mismatch_selector::operator() (
 _RAIter1 __i1,
 _RAIter2 __i2,
 _Pred __pred) [inline]
```

Test on one position.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__i1</code>   | _Iterator on first sequence.           |
| <code>__i2</code>   | _Iterator on second sequence (unused). |
| <code>__pred</code> | Find predicate.                        |

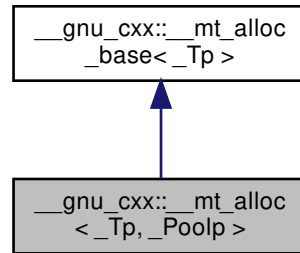
Definition at line 130 of file `find_selectors.h`.

The documentation for this struct was generated from the following file:

- [find\\_selectors.h](#)

4.40 `__gnu_cxx::__mt_alloc<_Tp, _Poolp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc<_Tp, _Poolp>`:



## Public Types

- `typedef _Poolp __policy_type`
- `typedef _Poolp::pool_type __pool_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true\_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `__mt_alloc (const \_\_mt\_alloc &) noexcept`
- `template<typename _Tp1, typename _Poolp1 >  
__mt_alloc (const \_\_mt\_alloc<_Tp1, _Poolp1> &) noexcept`
- `const __pool_base::_Tune M_get_options ()`
- `void M_set_options (__pool_base::_Tune __t)`
- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`
- `pointer allocate (size_type __n, const void * = 0)`
- `template<typename _Up, typename... _Args>  
void construct (_Up * __p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n)`
- `template<typename _Up >  
void destroy (_Up * __p)`
- `size_type max_size () const noexcept`

#### 4.40.1 Detailed Description

```
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
```

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).

Further details: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt\\_allocator.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html).

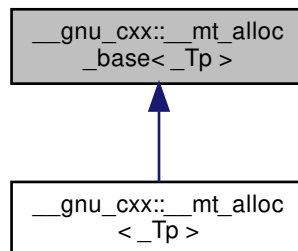
Definition at line 639 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 4.41 \_\_gnu\_cxx::\_\_mt\_alloc\_base< \_Tp > Class Template Reference

Inheritance diagram for `__gnu_cxx::__mt_alloc_base< _Tp >`:



#### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

##### 4.41.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__mt_alloc_base< _Tp >
```

Base class for \_Tp dependent member functions.

Definition at line 570 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

#### 4.42 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, ↵ _DifferenceTp, _Compare >` Struct Template Reference

#### Public Member Functions

- \_RAIter3 **operator**() (\_RAIterlterator \_\_seqs\_begin, \_RAIterlterator \_\_seqs\_end, \_RAIter3 \_\_target, ↵ \_DifferenceTp \_\_length, \_Compare \_\_comp)

##### 4.42.1 Detailed Description

```
template<bool __sentinels, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 3-way merging with \_\_sentinels turned off.

Note that 3-way merging is always stable!

Definition at line 752 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.43 `__gnu_parallel::__multiway_merge_3_variant_sentinel_switch` < true, `_RAIterIterator`, `_RAIter3`, `__DifferenceTp`, `_Compare` > Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterIterator` \_\_seqs\_begin, `_RAIterIterator` \_\_seqs\_end, `_RAIter3` \_\_target, `__DifferenceTp` \_\_length, `_Compare` \_\_comp)

##### 4.43.1 Detailed Description

```
template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 3-way merging with \_\_sentinels turned on.

Note that 3-way merging is always stable!

Definition at line 772 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.44 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch` < \_\_sentinels, `_RAIterIterator`, `_RAIter3`, `__DifferenceTp`, `_Compare` > Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterIterator` \_\_seqs\_begin, `_RAIterIterator` \_\_seqs\_end, `_RAIter3` \_\_target, `__DifferenceTp` \_\_length, `_Compare` \_\_comp)

##### 4.44.1 Detailed Description

```
template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare
>
```

Switch for 4-way merging with \_\_sentinels turned off.

Note that 4-way merging is always stable!

Definition at line 795 of file multiway\_merge.h.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.45 `__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, `__DifferenceTp __length`, `_Compare __comp`)

##### 4.45.1 Detailed Description

```
template<typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for 4-way merging with `__sentinels` turned on.

Note that 4-way merging is always stable!

Definition at line 815 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.46 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator()` (`_RAIterlterator __seqs_begin`, `_RAIterlterator __seqs_end`, `_RAIter3 __target`, const typename `std::iterator_traits< typename std::iterator_traits< _RAIterlterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp`)

##### 4.46.1 Detailed Description

```
template<bool __sentinels, bool __stable, typename _RAIterlterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterlterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned on.

Definition at line 837 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.47 `__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >` Struct Template Reference

##### Public Member Functions

- `_RAIter3 operator() (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type & __sentinel, _DifferenceTp __length, _Compare __comp)`

##### 4.47.1 Detailed Description

```
template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
```

Switch for k-way merging with `__sentinels` turned off.

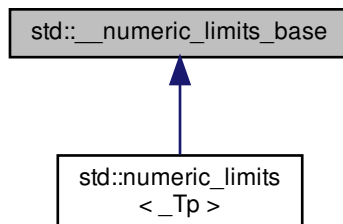
Definition at line 872 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.48 `std::__numeric_limits_base` Struct Reference

Inheritance diagram for `std::__numeric_limits_base`:





## Static Public Attributes

- static constexpr int [digits](#)
- static constexpr int [digits10](#)
- static constexpr [float\\_denorm\\_style](#) [has\\_denorm](#)
- static constexpr bool [has\\_denorm\\_loss](#)
- static constexpr bool [has\\_infinity](#)
- static constexpr bool [has\\_quiet\\_NaN](#)
- static constexpr bool [has\\_signaling\\_NaN](#)
- static constexpr bool [is\\_bounded](#)
- static constexpr bool [is\\_exact](#)
- static constexpr bool [is\\_iec559](#)
- static constexpr bool [is\\_integer](#)
- static constexpr bool [is\\_modulo](#)
- static constexpr bool [is\\_signed](#)
- static constexpr bool [is\\_specialized](#)
- static constexpr int [max\\_digits10](#)
- static constexpr int [max\\_exponent](#)
- static constexpr int [max\\_exponent10](#)
- static constexpr int [min\\_exponent](#)
- static constexpr int [min\\_exponent10](#)
- static constexpr int [radix](#)
- static constexpr [float\\_round\\_style](#) [round\\_style](#)
- static constexpr bool [tinyness\\_before](#)
- static constexpr bool [traps](#)

## 4.48.1 Detailed Description

Part of `std::numeric_limits`.

The `static const` members are usable as integral constant expressions.

## Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the `std::numeric_limits` class.

Definition at line 202 of file `limits`.

## 4.48.2 Member Data Documentation

#### 4.48.2.1 digits

```
constexpr int std::__numeric_limits_base::digits [static]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 211 of file `limits`.

#### 4.48.2.2 digits10

```
constexpr int std::__numeric_limits_base::digits10 [static]
```

The number of base 10 digits that can be represented without change.

Definition at line 214 of file `limits`.

#### 4.48.2.3 has\_denorm

```
constexpr float_denorm_style std::__numeric_limits_base::has_denorm [static]
```

See `std::float_denorm_style` for more information.

Definition at line 266 of file `limits`.

#### 4.48.2.4 has\_denorm\_loss

```
constexpr bool std::__numeric_limits_base::has_denorm_loss [static]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 270 of file `limits`.

#### 4.48.2.5 has\_infinity

```
constexpr bool std::__numeric_limits_base::has_infinity [static]
```

True if the type has a representation for positive infinity.

Definition at line 255 of file `limits`.

#### 4.48.2.6 `has_quiet_NaN`

```
constexpr bool std::__numeric_limits_base::has_quiet_NaN [static]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 259 of file `limits`.

#### 4.48.2.7 `has_signaling_NaN`

```
constexpr bool std::__numeric_limits_base::has_signaling_NaN [static]
```

True if the type has a representation for a signaling Not a Number.

Definition at line 263 of file `limits`.

#### 4.48.2.8 `is_bounded`

```
constexpr bool std::__numeric_limits_base::is_bounded [static]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 279 of file `limits`.

#### 4.48.2.9 `is_exact`

```
constexpr bool std::__numeric_limits_base::is_exact [static]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 231 of file `limits`.

#### 4.48.2.10 `is_iec559`

```
constexpr bool std::__numeric_limits_base::is_iec559 [static]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 274 of file `limits`.

#### 4.48.2.11 `is_integer`

```
constexpr bool std::__numeric_limits_base::is_integer [static]
```

True if the type is integer.

Definition at line 226 of file limits.

#### 4.48.2.12 `is_modulo`

```
constexpr bool std::__numeric_limits_base::is_modulo [static]
```

True if the type is *modulo*. A type is modulo if, for any operation involving `+`, `-`, or `*` on values of that type whose result would fall outside the range `[min(),max()]`, the value returned differs from the true value by an integer multiple of `max() - min() + 1`. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 288 of file limits.

#### 4.48.2.13 `is_signed`

```
constexpr bool std::__numeric_limits_base::is_signed [static]
```

True if the type is signed.

Definition at line 223 of file limits.

#### 4.48.2.14 `is_specialized`

```
constexpr bool std::__numeric_limits_base::is_specialized [static]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 206 of file limits.

#### 4.48.2.15 `max_digits10`

```
constexpr int std::__numeric_limits_base::max_digits10 [static]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 219 of file limits.

#### 4.48.2.16 `max_exponent`

```
constexpr int std::numeric_limits_base::max_exponent [static]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 248 of file `limits`.

#### 4.48.2.17 `max_exponent10`

```
constexpr int std::numeric_limits_base::max_exponent10 [static]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 252 of file `limits`.

#### 4.48.2.18 `min_exponent`

```
constexpr int std::numeric_limits_base::min_exponent [static]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 239 of file `limits`.

#### 4.48.2.19 `min_exponent10`

```
constexpr int std::numeric_limits_base::min_exponent10 [static]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 243 of file `limits`.

#### 4.48.2.20 `radix`

```
constexpr int std::numeric_limits_base::radix [static]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 235 of file `limits`.

#### 4.48.2.21 round\_style

```
constexpr float_round_style std::__numeric_limits_base::round_style [static]
```

See `std::float_round_style` for more information. This is only meaningful for floating types; integer types will all be `round_toward_zero`.

Definition at line 299 of file `limits`.

#### 4.48.2.22 tinyness\_before

```
constexpr bool std::__numeric_limits_base::tinyness_before [static]
```

True if tininess is detected before rounding. (see IEC 559)

Definition at line 294 of file `limits`.

#### 4.48.2.23 traps

```
constexpr bool std::__numeric_limits_base::traps [static]
```

True if trapping is implemented for this type.

Definition at line 291 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

### 4.49 \_\_gnu\_cxx::\_\_per\_type\_pool\_policy<\_Tp, \_PoolTp, \_Thread> Struct Template Reference

Inherits `__gnu_cxx::__per_type_pool_base<_Tp, _PoolTp, _Thread>`.

#### 4.49.1 Detailed Description

```
template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy<_Tp, _PoolTp, _Thread>
```

Policy for individual `__pool` objects.

Definition at line 555 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

## 4.50 \_\_gnu\_cxx::\_\_pool&lt;\_Thread&gt; Class Template Reference

## 4.50.1 Detailed Description

```
template<bool _Thread>
class __gnu_cxx::__pool<_Thread>
```

Data describing the underlying memory pool, parameterized on threading support.

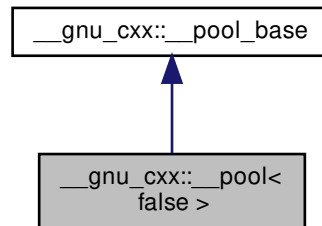
Definition at line 191 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 4.51 \_\_gnu\_cxx::\_\_pool&lt;false&gt; Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool<false>`:



## Public Types

- typedef unsigned short int **\_Binmap\_type**
- typedef std::size\_t **size\_t**

## Public Member Functions

- **\_\_pool** (const \_\_pool\_base:: \_Tune &\_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record &, \_Block\_record \*, size\_t)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \* \_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

## Protected Attributes

- `_Binmap_type * _M_binmap`
- `bool _M_init`
- `_Tune _M_options`

### 4.51.1 Detailed Description

```
template<>
class __gnu_cxx::__pool< false >
```

Specialization for single thread.

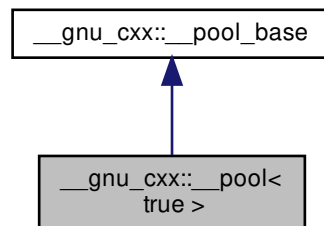
Definition at line 195 of file `mt_allocator.h`.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

### 4.52 `__gnu_cxx::__pool< true >` Class Template Reference

Inheritance diagram for `__gnu_cxx::__pool< true >`:



## Public Types

- `typedef unsigned short int _Binmap_type`
- `typedef std::size_t size_t`



## Public Member Functions

- **\_\_pool** (const \_\_pool\_base:: \_Tune & \_\_tune)
- void **\_M\_adjust\_freelist** (const \_Bin\_record & \_\_bin, \_Block\_record \* \_\_block, size\_t \_\_thread\_id)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- void **\_M\_destroy** () throw ()
- void **\_M\_destroy\_thread\_key** (void \*) throw ()
- size\_t **\_M\_get\_align** ()
- const \_Bin\_record & **\_M\_get\_bin** (size\_t \_\_which)
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- size\_t **\_M\_get\_thread\_id** ()
- void **\_M\_initialize** (\_\_destroy\_handler)
- void **\_M\_initialize\_once** ()
- void **\_M\_reclaim\_block** (char \* \_\_p, size\_t \_\_bytes) throw ()
- char \* **\_M\_reserve\_block** (size\_t \_\_bytes, const size\_t \_\_thread\_id)
- void **\_M\_set\_options** (\_Tune \_\_t)

## Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

## 4.52.1 Detailed Description

```
template<>
class __gnu_cxx::__pool< true >
```

Specialization for thread enabled, via gthreads.h.

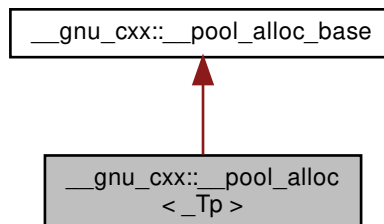
Definition at line 262 of file mt\_allocator.h.

The documentation for this class was generated from the following file:

- [mt\\_allocator.h](#)

## 4.53 \_\_gnu\_cxx::\_\_pool\_alloc&lt;\_Tp&gt; Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc<\_Tp>:



### Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#) &) noexcept
- template<typename \_Tp1 >  
  **\_\_pool\_alloc** (const [\\_\\_pool\\_alloc](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*=0)
- template<typename \_Up, typename... \_Args>  
  void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
  void **destroy** (\_Up \*\_\_p)
- size\_type **max\_size** () const noexcept

### Private Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

### Private Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Private Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

## 4.53.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__pool_alloc< _Tp >
```

Allocator using a memory pool with a single lock.

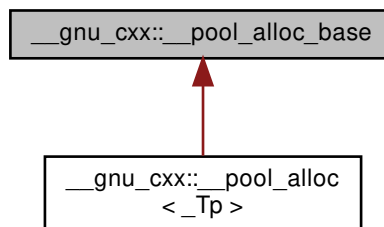
Definition at line 124 of file pool\_allocator.h.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

## 4.54 \_\_gnu\_cxx::\_\_pool\_alloc\_base Class Reference

Inheritance diagram for \_\_gnu\_cxx::\_\_pool\_alloc\_base:



## Protected Types

- enum { **\_S\_align** }
- enum { **\_S\_max\_bytes** }
- enum { **\_S\_free\_list\_size** }

## Protected Member Functions

- char \* **\_M\_allocate\_chunk** (size\_t \_\_n, int &\_\_nobjs)
- \_Obj \*volatile \* **\_M\_get\_free\_list** (size\_t \_\_bytes) throw ()
- \_\_mutex & **\_M\_get\_mutex** () throw ()
- void \* **\_M\_refill** (size\_t \_\_n)
- size\_t **\_M\_round\_up** (size\_t \_\_bytes)

### Static Protected Attributes

- static char \* **\_S\_end\_free**
- static \_Obj \*volatile **\_S\_free\_list** [\_S\_free\_list\_size]
- static size\_t **\_S\_heap\_size**
- static char \* **\_S\_start\_free**

#### 4.54.1 Detailed Description

Base class for `__pool_alloc`.

Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).

Important implementation properties: 0. If globally mandated, then allocate objects from new

1. If the clients request an object of size  $> \_S\_max\_bytes$ , the resulting object will be obtained directly from new
2. In all other cases, we allocate an object of size exactly `\_S_round_up(requested_size)`. Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

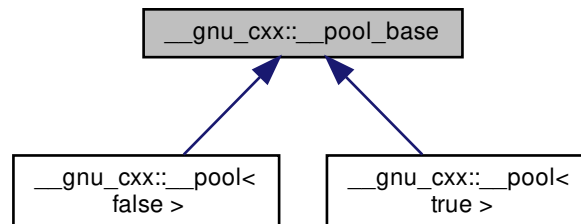
Definition at line 75 of file `pool_allocator.h`.

The documentation for this class was generated from the following file:

- [pool\\_allocator.h](#)

#### 4.55 `__gnu_cxx::__pool_base` Struct Reference

Inheritance diagram for `__gnu_cxx::__pool_base`:



## Public Types

- typedef unsigned short int **\_Binmap\_type**
- typedef std::size\_t **size\_t**

## Public Member Functions

- **\_\_pool\_base** (const \_Tune &\_\_options)
- bool **\_M\_check\_threshold** (size\_t \_\_bytes)
- size\_t **\_M\_get\_align** ()
- size\_t **\_M\_get\_binmap** (size\_t \_\_bytes)
- const \_Tune & **\_M\_get\_options** () const
- void **\_M\_set\_options** (\_Tune \_\_t)

## Protected Attributes

- \_Binmap\_type \* **\_M\_binmap**
- bool **\_M\_init**
- \_Tune **\_M\_options**

## 4.55.1 Detailed Description

Base class for pool object.

Definition at line 49 of file `mt_allocator.h`.

The documentation for this struct was generated from the following file:

- [mt\\_allocator.h](#)

4.56 `__gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::__vstring_utility<_CharT, _Traits, _Alloc>`.

## Public Types

- typedef \_Util\_Base::\_CharT\_alloc\_type **\_CharT\_alloc\_type**
- typedef \_\_vstring\_utility<\_CharT, \_Traits, \_Alloc> **\_Util\_Base**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT\_alloc\_type::size\_type **size\_type**
- typedef \_Traits **traits\_type**
- typedef \_Traits::char\_type **value\_type**

## Public Member Functions

- `__rc_string_base` (const `_Alloc` & `_a`)
- `__rc_string_base` (const `__rc_string_base` & `_rcs`)
- `__rc_string_base` (`__rc_string_base` && `_rcs`)
- `__rc_string_base` (size\_type `__n`, `_CharT` `__c`, const `_Alloc` & `_a`)
- `template<typename _InputIterator >`  
`__rc_string_base` (`_InputIterator` `__beg`, `_InputIterator` `__end`, const `_Alloc` & `_a`)
- `void` `_M_assign` (const `__rc_string_base` & `_rcs`)
- `size_type` `_M_capacity` () const
- `void` `_M_clear` ()
- `bool` `_M_compare` (const `__rc_string_base` &) const
- `template<>`  
`bool` `_M_compare` (const `__rc_string_base` & `_rcs`) const
- `template<>`  
`bool` `_M_compare` (const `__rc_string_base` & `_rcs`) const
- `_CharT *` `_M_data` () const
- `void` `_M_erase` (size\_type `__pos`, size\_type `__n`)
- `allocator_type` & `_M_get_allocator` ()
- `const allocator_type` & `_M_get_allocator` () const
- `bool` `_M_is_shared` () const
- `void` `_M_leak` ()
- `size_type` `_M_length` () const
- `size_type` `_M_max_size` () const
- `void` `_M_mutate` (size\_type `__pos`, size\_type `__len1`, const `_CharT *` `__s`, size\_type `__len2`)
- `void` `_M_reserve` (size\_type `__res`)
- `void` `_M_set_leaked` ()
- `void` `_M_set_length` (size\_type `__n`)
- `void` `_M_swap` (`__rc_string_base` & `_rcs`)
- `template<typename _InIterator >`  
`_CharT *` `_S_construct` (`_InIterator` `__beg`, `_InIterator` `__end`, const `_Alloc` & `_a`, `std::forward_iterator_tag`)

## Protected Types

- `typedef` `__gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > >` `__const_rc_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_string_base > >` `__const_sso_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > >` `__rc_iterator`
- `typedef` `__gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_string_base > >` `__sso_iterator`
- `typedef` `__alloc_traits< _CharT_alloc_type >` `_CharT_alloc_traits`
- `typedef` `_CharT_alloc_traits::const_pointer` `const_pointer`
- `typedef` `_CharT_alloc_type::difference_type` `difference_type`
- `typedef` `_CharT_alloc_traits::pointer` `pointer`

## Static Protected Member Functions

- static void `_S_assign` (`_CharT *__d`, `size_type __n`, `_CharT __c`)
- static int `_S_compare` (`size_type __n1`, `size_type __n2`)
- static void `_S_copy` (`_CharT *__d`, `const _CharT *__s`, `size_type __n`)
- template<typename `_Iterator`>  
static void `_S_copy_chars` (`_CharT *__p`, `_Iterator __k1`, `_Iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `_sso_iterator __k1`, `_sso_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `_const_sso_iterator __k1`, `_const_sso_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `_rc_iterator __k1`, `_rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `_const_rc_iterator __k1`, `_const_rc_iterator __k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `_CharT *__k1`, `_CharT *__k2`)
- static void `_S_copy_chars` (`_CharT *__p`, `const _CharT *__k1`, `const _CharT *__k2`)
- static void `_S_move` (`_CharT *__d`, `const _CharT *__s`, `size_type __n`)

## 4.56.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base<_CharT, _Traits, _Alloc>
```

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

 [_Rep]
 _M_length
[__rc_string_base<char_type>] _M_capacity
_M_dataplus _M_refcount
_M_p -----> unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_refdata()`, and `__rc_string_base::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 83 of file `rc_string_base.h`.

The documentation for this class was generated from the following file:

- [rc\\_string\\_base.h](#)

## 4.57 `std::tr2::__reflection_typelist<_Elements>` Struct Template Reference

### 4.57.1 Detailed Description

```
template<typename... _Elements>
struct std::tr2::__reflection_typelist<_Elements>
```

See N2965: Type traits and base classes by Michael SpertusSimple typelist. Compile-time list of types.

Definition at line 56 of file tr2/type\_traits.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 4.58 `std::tr2::__reflection_typelist<_First, _Rest...>` Struct Template Reference

### Public Types

- typedef [std::false\\_type](#) `empty`

### 4.58.1 Detailed Description

```
template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist<_First, _Rest...>
```

Partial specialization.

Definition at line 67 of file tr2/type\_traits.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

## 4.59 `std::tr2::__reflection_typelist<>` Struct Template Reference

### Public Types

- typedef [std::true\\_type](#) `empty`



## 4.59.1 Detailed Description

```
template<>
struct std::tr2::__reflection_typelist<>
```

Specialization for an empty typelist.

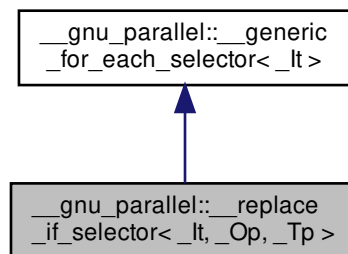
Definition at line 60 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

4.60 `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`:



## Public Member Functions

- [\\_\\_replace\\_if\\_selector](#) (const `_Tp` & [\\_\\_new\\_val](#))
- bool [operator\(\)](#) (`_Op` & `__o`, `_It` `__i`)

## Public Attributes

- const `_Tp` & [\\_\\_new\\_val](#)
- `_It` [\\_M\\_finish\\_iterator](#)

#### 4.60.1 Detailed Description

```
template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >
```

std::replace() selector.

Definition at line 156 of file for\_each\_selectors.h.

#### 4.60.2 Constructor & Destructor Documentation

##### 4.60.2.1 \_\_replace\_if\_selector()

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
 const _Tp & __new_val) [inline], [explicit]
```

Constructor.

##### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

Definition at line 164 of file for\_each\_selectors.h.

#### 4.60.3 Member Function Documentation

##### 4.60.3.1 operator>()()

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

Definition at line 170 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val`.

#### 4.60.4 Member Data Documentation

##### 4.60.4.1 `__new_val`

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::__new_val
```

Value to replace with.

Definition at line 159 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>::operator()`.

##### 4.60.4.2 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
```

`_Iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

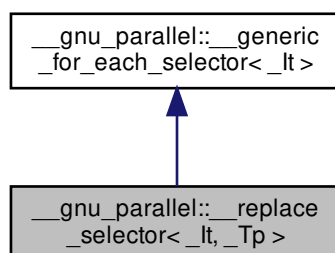
Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.61 `__gnu_parallel::__replace_selector<_It, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::__replace_selector<_It, _Tp>`:



### Public Member Functions

- [\\_\\_replace\\_selector](#) (const \_Tp &\_\_new\_val)
- bool [operator\(\)](#) (\_Tp &\_\_v, \_It \_\_i)

### Public Attributes

- const \_Tp & [\\_\\_new\\_val](#)
- \_It [\\_M\\_finish\\_iterator](#)

#### 4.61.1 Detailed Description

```
template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector<_It, _Tp>
```

std::replace() selector.

Definition at line 132 of file for\_each\_selectors.h.

#### 4.61.2 Constructor & Destructor Documentation

##### 4.61.2.1 \_\_replace\_selector()

```
template<typename _It , typename _Tp >
__gnu_parallel::__replace_selector<_It, _Tp>::__replace_selector (
 const _Tp & __new_val) [inline], [explicit]
```

Constructor.

#### Parameters

|                        |                        |
|------------------------|------------------------|
| <code>__new_val</code> | Value to replace with. |
|------------------------|------------------------|

Definition at line 140 of file for\_each\_selectors.h.

#### 4.61.3 Member Function Documentation

4.61.3.1 `operator()`

```
template<typename _It , typename _Tp >
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
 _Tp & __v,
 _It __i) [inline]
```

Functor execution.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__v</code> | Current value.               |
| <code>__i</code> | iterator referencing object. |

Definition at line 146 of file `for_each_selectors.h`.

References `__gnu_parallel::__replace_selector<_It, _Tp>::__new_val`.

## 4.61.4 Member Data Documentation

4.61.4.1 `__new_val`

```
template<typename _It , typename _Tp >
const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val
```

Value to replace with.

Definition at line 135 of file `for_each_selectors.h`.

Referenced by `__gnu_parallel::__replace_selector<_It, _Tp>::operator()`.

4.61.4.2 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

Iterator on last element processed; needed for some algorithms (e. g. `std::transform()`).

Definition at line 47 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.62 \_\_gnu\_cxx::\_\_scoped\_lock Class Reference

### Public Types

- typedef \_\_mutex \_\_mutex\_type

### Public Member Functions

- **\_\_scoped\_lock** (\_\_mutex\_type &\_\_name)

### 4.62.1 Detailed Description

Scoped lock idiom.

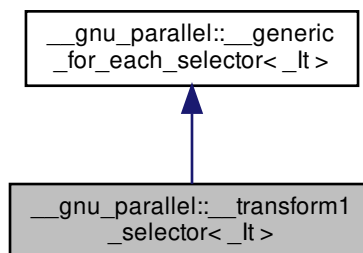
Definition at line 228 of file `concurrency.h`.

The documentation for this class was generated from the following file:

- [concurrency.h](#)

## 4.63 \_\_gnu\_parallel::\_\_transform1\_selector<\_It> Struct Template Reference

Inheritance diagram for `__gnu_parallel::__transform1_selector<_It>`:



### Public Member Functions

- template<typename \_Op >  
bool [operator\(\)](#) (\_Op &\_\_o, \_It \_\_i)

## Public Attributes

- [\\_It \\_M\\_finish\\_iterator](#)

## 4.63.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform1_selector<_It>
```

std::transform() \_\_selector, one input sequence variant.

Definition at line 100 of file for\_each\_selectors.h.



## 4.63.2 Member Function Documentation

## 4.63.2.1 operator&gt;()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform1_selector<_It>::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

## Parameters

|                                                                                                       |                              |
|-------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#">_o</a> | Operator.                    |
| <a href="#">_i</a> | iterator referencing object. |

Definition at line 107 of file for\_each\_selectors.h.

## 4.63.3 Member Data Documentation

## 4.63.3.1 \_M\_finish\_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector<_It>::_M_finish_iterator [inherited]
```

\_\_iterator on last element processed; needed for some algorithms (e. g. std::transform()).

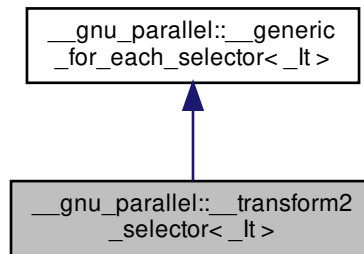
Definition at line 47 of file for\_each\_selectors.h.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.64 \_\_gnu\_parallel::\_\_transform2\_selector<\_It> Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_\_transform2\_selector<\_It>:



##### Public Member Functions

- `template<typename _Op >`  
`bool operator\(\) (_Op &__o, _It __i)`

##### Public Attributes

- `_It \_M\_finish\_iterator`

##### 4.64.1 Detailed Description

```
template<typename _It>
struct __gnu_parallel::__transform2_selector<_It>
```

std::transform() \_\_selector, two input sequences variant.

Definition at line 116 of file for\_each\_selectors.h.



## 4.64.2 Member Function Documentation

4.64.2.1 `operator()`

```
template<typename _It>
template<typename _Op >
bool __gnu_parallel::__transform2_selector< _It >::operator() (
 _Op & __o,
 _It __i) [inline]
```

Functor execution.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__o</code> | Operator.                    |
| <code>__i</code> | iterator referencing object. |

Definition at line 123 of file `_each_selectors.h`.

## 4.64.3 Member Data Documentation

4.64.3.1 `_M_finish_iterator`

```
template<typename _It >
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator [inherited]
```

`_M_finish_iterator` on last element processed; needed for some algorithms (e. g. `std::transform()`).

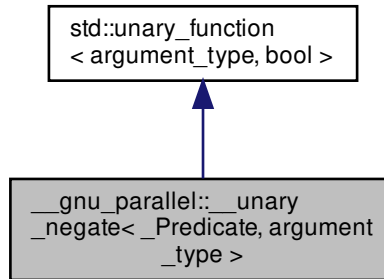
Definition at line 47 of file `_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [\\_each\\_selectors.h](#)

#### 4.65 `__gnu_parallel::__unary_negate<_Predicate, argument_type>` Class Template Reference

Inheritance diagram for `__gnu_parallel::__unary_negate<_Predicate, argument_type>`:



##### Public Types

- typedef [argument\\_type](#) `argument_type`
- typedef bool [result\\_type](#)

##### Public Member Functions

- **`__unary_negate`** (const `_Predicate` &`__x`)
- bool **`operator()`** (const [argument\\_type](#) &`__x`)

##### Protected Attributes

- `_Predicate _M_pred`

##### 4.65.1 Detailed Description

```
template<typename _Predicate, typename argument_type>
class __gnu_parallel::__unary_negate<_Predicate, argument_type>
```

Similar to `std::unary_negate`, but giving the argument types explicitly.

Definition at line 173 of file `base.h`.

##### 4.65.2 Member Typedef Documentation

4.65.2.1 `argument_type`

```
typedef argument_type std::unary_function< argument_type , bool >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.65.2.2 `result_type`

```
typedef bool std::unary_function< argument_type , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [base.h](#)

4.66 `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >` Class Template Reference

Inherits `_Base<_CharT, _Traits, _Alloc >`.

## Public Types

- typedef `_Alloc` **`allocator_type`**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, __versa_string >` **`const_iterator`**
- typedef `_CharT_alloc_traits::const_pointer` **`const_pointer`**
- typedef `const value_type &` **`const_reference`**
- typedef `std::reverse_iterator< const_iterator >` **`const_reverse_iterator`**
- typedef `_CharT_alloc_type::difference_type` **`difference_type`**
- typedef `__gnu_cxx::__normal_iterator< pointer, __versa_string >` **`iterator`**
- typedef `_CharT_alloc_traits::pointer` **`pointer`**
- typedef `value_type &` **`reference`**
- typedef `std::reverse_iterator< iterator >` **`reverse_iterator`**
- typedef `_CharT_alloc_type::size_type` **`size_type`**
- typedef `_Traits` **`traits_type`**
- typedef `_Traits::char_type` **`value_type`**

## Public Member Functions

- [\\_\\_versa\\_string](#) (const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#)) noexcept
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#))
- [\\_\\_versa\\_string](#) ([\\_\\_versa\\_string](#) && [\\_\\_str](#)) noexcept
- [\\_\\_versa\\_string](#) (std::initializer\_list< [\\_CharT](#) > [\\_\\_l](#), const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#))
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#), size\_type [\\_\\_pos](#), size\_type [\\_\\_n](#)=npos)
- [\\_\\_versa\\_string](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#), size\_type [\\_\\_pos](#), size\_type [\\_\\_n](#), const [\\_Alloc](#) & [\\_\\_a](#))
- [\\_\\_versa\\_string](#) (const [\\_CharT](#) \* [\\_\\_s](#), size\_type [\\_\\_n](#), const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#))
- [\\_\\_versa\\_string](#) (const [\\_CharT](#) \* [\\_\\_s](#), const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#))
- [\\_\\_versa\\_string](#) (size\_type [\\_\\_n](#), [\\_CharT](#) [\\_\\_c](#), const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#))
- template<class [\\_InputIterator](#) , typename = std::RequireInputIter< [\\_InputIterator](#) >>  
  [\\_\\_versa\\_string](#) ([\\_InputIterator](#) [\\_\\_beg](#), [\\_InputIterator](#) [\\_\\_end](#), const [\\_Alloc](#) & [\\_\\_a](#)=[\\_Alloc\(\)](#))
- [~\\_\\_versa\\_string](#) () noexcept
- template<typename [\\_InputIterator](#) >  
  [\\_\\_versa\\_string](#)< [\\_CharT](#), [\\_Traits](#), [\\_Alloc](#), [\\_Base](#) > & **[M\\_replace\\_dispatch](#)** (const\_iterator [\\_\\_i1](#), const\_iterator [\\_\\_i2](#), [\\_InputIterator](#) [\\_\\_k1](#), [\\_InputIterator](#) [\\_\\_k2](#), std::false\_type)
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#))
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#), size\_type [\\_\\_pos](#), size\_type [\\_\\_n](#))
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_CharT](#) \* [\\_\\_s](#), size\_type [\\_\\_n](#))
- [\\_\\_versa\\_string](#) & [append](#) (const [\\_CharT](#) \* [\\_\\_s](#))
- [\\_\\_versa\\_string](#) & [append](#) (size\_type [\\_\\_n](#), [\\_CharT](#) [\\_\\_c](#))
- [\\_\\_versa\\_string](#) & [append](#) (std::initializer\_list< [\\_CharT](#) > [\\_\\_l](#))
- template<class [\\_InputIterator](#) , typename = std::RequireInputIter< [\\_InputIterator](#) >>  
  [\\_\\_versa\\_string](#) & [append](#) ([\\_InputIterator](#) [\\_\\_first](#), [\\_InputIterator](#) [\\_\\_last](#))
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#))
- [\\_\\_versa\\_string](#) & [assign](#) ([\\_\\_versa\\_string](#) && [\\_\\_str](#)) noexcept
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#), size\_type [\\_\\_pos](#), size\_type [\\_\\_n](#))
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_CharT](#) \* [\\_\\_s](#), size\_type [\\_\\_n](#))
- [\\_\\_versa\\_string](#) & [assign](#) (const [\\_CharT](#) \* [\\_\\_s](#))
- [\\_\\_versa\\_string](#) & [assign](#) (size\_type [\\_\\_n](#), [\\_CharT](#) [\\_\\_c](#))
- template<class [\\_InputIterator](#) , typename = std::RequireInputIter< [\\_InputIterator](#) >>  
  [\\_\\_versa\\_string](#) & [assign](#) ([\\_InputIterator](#) [\\_\\_first](#), [\\_InputIterator](#) [\\_\\_last](#))
- [\\_\\_versa\\_string](#) & [assign](#) (std::initializer\_list< [\\_CharT](#) > [\\_\\_l](#))
- const\_reference [at](#) (size\_type [\\_\\_n](#)) const
- reference [at](#) (size\_type [\\_\\_n](#))
- reference [back](#) () noexcept
- const\_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- const [\\_CharT](#) \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- int [compare](#) (const [\\_\\_versa\\_string](#) & [\\_\\_str](#)) const
- int [compare](#) (size\_type [\\_\\_pos](#), size\_type [\\_\\_n](#), const [\\_\\_versa\\_string](#) & [\\_\\_str](#)) const
- int [compare](#) (size\_type [\\_\\_pos1](#), size\_type [\\_\\_n1](#), const [\\_\\_versa\\_string](#) & [\\_\\_str](#), size\_type [\\_\\_pos2](#), size\_type [\\_\\_n2](#)) const
- int [compare](#) (const [\\_CharT](#) \* [\\_\\_s](#)) const
- int [compare](#) (size\_type [\\_\\_pos](#), size\_type [\\_\\_n1](#), const [\\_CharT](#) \* [\\_\\_s](#)) const

- `int compare` (`size_type __pos`, `size_type __n1`, `const _CharT *__s`, `size_type __n2`) `const`
- `size_type copy` (`_CharT *__s`, `size_type __n`, `size_type __pos=0`) `const`
- `const_reverse_iterator crbegin` () `const noexcept`
- `const_reverse_iterator crend` () `const noexcept`
- `const _CharT * data` () `const noexcept`
- `bool empty` () `const noexcept`
- `iterator end` () `noexcept`
- `const_iterator end` () `const noexcept`
- `__versa_string & erase` (`size_type __pos=0`, `size_type __n=npos`)
- `iterator erase` (`const_iterator __position`)
- `iterator erase` (`const_iterator __first`, `const_iterator __last`)
- `size_type find` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_not_of` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_not_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_not_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const __versa_string & __str`, `size_type __pos=0`) `const noexcept`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_first_of` (`const _CharT *__s`, `size_type __pos=0`) `const`
- `size_type find_first_of` (`_CharT __c`, `size_type __pos=0`) `const noexcept`
- `size_type find_last_not_of` (`const __versa_string & __str`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_not_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `size_type find_last_not_of` (`_CharT __c`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_of` (`const __versa_string & __str`, `size_type __pos=npos`) `const noexcept`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos`, `size_type __n`) `const`
- `size_type find_last_of` (`const _CharT *__s`, `size_type __pos=npos`) `const`
- `size_type find_last_of` (`_CharT __c`, `size_type __pos=npos`) `const noexcept`
- `reference front` () `noexcept`
- `const_reference front` () `const noexcept`
- `allocator_type get_allocator` () `const noexcept`
- `iterator insert` (`const_iterator __p`, `size_type __n`, `_CharT __c`)
- `template<class InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`iterator insert` (`const_iterator __p`, `InputIterator __beg`, `InputIterator __end`)
- `iterator insert` (`const_iterator __p`, `std::initializer_list<_CharT> __l`)
- `__versa_string & insert` (`size_type __pos1`, `const __versa_string & __str`)
- `__versa_string & insert` (`size_type __pos1`, `const __versa_string & __str`, `size_type __pos2`, `size_type __n`)
- `__versa_string & insert` (`size_type __pos`, `const _CharT *__s`, `size_type __n`)
- `__versa_string & insert` (`size_type __pos`, `const _CharT *__s`)
- `__versa_string & insert` (`size_type __pos`, `size_type __n`, `_CharT __c`)
- `iterator insert` (`const_iterator __p`, `_CharT __c`)
- `size_type length` () `const noexcept`
- `size_type max_size` () `const noexcept`
- `__versa_string & operator+=` (`const __versa_string & __str`)
- `__versa_string & operator+=` (`const _CharT *__s`)
- `__versa_string & operator+=` (`_CharT __c`)
- `__versa_string & operator+=` (`std::initializer_list<_CharT> __l`)
- `__versa_string & operator=` (`const __versa_string & __str`)

- `__versa_string & operator= (__versa_string &&__str)` noexcept
- `__versa_string & operator= (std::initializer_list<_CharT> __l)`
- `__versa_string & operator= (const _CharT *__s)`
- `__versa_string & operator= (_CharT __c)`
- `const_reference operator[] (size_type __pos)` const noexcept
- `reference operator[] (size_type __pos)` noexcept
- `void pop_back ()`
- `void push_back (_CharT __c)`
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- `__versa_string & replace (size_type __pos, size_type __n, const __versa_string &__str)`
- `__versa_string & replace (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)`
- `__versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s)`
- `__versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const __versa_string &__str)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s, size_type __n)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c)`
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`__versa_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, _CharT *__k1, _CharT *__k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__k1, const _CharT *__k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2)`
- `__versa_string & replace (const_iterator __i1, const_iterator __i2, std::initializer_list<_CharT> __l)`
- `void reserve (size_type __res_arg=0)`
- `void resize (size_type __n, _CharT __c)`
- `void resize (size_type __n)`
- `size_type rfind (const __versa_string &__str, size_type __pos=npos)` const noexcept
- `size_type rfind (const _CharT *__s, size_type __pos, size_type __n)` const
- `size_type rfind (const _CharT *__s, size_type __pos=npos)` const
- `size_type rfind (_CharT __c, size_type __pos=npos)` const noexcept
- `void shrink_to_fit ()` noexcept
- `size_type size ()` const noexcept
- `__versa_string substr (size_type __pos=0, size_type __n=npos)` const
- `void swap (__versa_string &__s)` noexcept

#### Static Public Attributes

- static const size\_type `npos`

## 4.66.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
```

Template class `__versa_string`.

Data structure managing sequences of characters and character-like objects.

Definition at line 56 of file `vstring.h`.

## 4.66.2 Constructor &amp; Destructor Documentation

4.66.2.1 `__versa_string()` [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const _Alloc & __a = _Alloc()) [inline], [explicit], [noexcept]
```

Construct an empty string using allocator `a`.

Definition at line 138 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr()`.

4.66.2.2 `__versa_string()` [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Construct string with copy of value of `__str`.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

Definition at line 146 of file `vstring.h`.

#### 4.66.2.3 `__versa_string()` [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 __versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

String move constructor.

##### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-constructed string contains the exact contents of `__str`. The contents of `__str` are a valid, but unspecified string.

Definition at line 158 of file `vstring.h`.

#### 4.66.2.4 `__versa_string()` [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 std::initializer_list< _CharT > __l,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string from an initializer list.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__l</code> | <code>std::initializer_list</code> of characters. |
| <code>__a</code> | Allocator to use (default is default allocator).  |

Definition at line 166 of file `vstring.h`.

#### 4.66.2.5 `__versa_string()` [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n = npos) [inline]
```

Construct string as copy of a substring.



## Parameters

|                    |                                                   |
|--------------------|---------------------------------------------------|
| <code>__str</code> | Source string.                                    |
| <code>__pos</code> | Index of first character to copy from.            |
| <code>__n</code>   | Number of characters to copy (default remainder). |

Definition at line 177 of file `vstring.h`.

4.66.2.6 `__versa_string()` [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n,
 const _Alloc & __a) [inline]
```

Construct string as copy of a substring.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

Definition at line 192 of file `vstring.h`.

4.66.2.7 `__versa_string()` [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const _CharT * __s,
 size_type __n,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string initialized by a character array.

## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'0'` has no special meaning.

Definition at line 209 of file `vstring.h`.

#### 4.66.2.8 `__versa_string()` [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 const _CharT * __s,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a C string.

##### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source C string.                                 |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 218 of file `vstring.h`.

#### 4.66.2.9 `__versa_string()` [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 size_type __n,
 _CharT __c,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as multiple characters.

##### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__n</code> | Number of characters.                            |
| <code>__c</code> | Character to use.                                |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 228 of file `vstring.h`.

4.66.2.10 `__versa_string()` [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
 _InputIterator __beg,
 _InputIterator __end,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a range.

## Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

Definition at line 243 of file `vstring.h`.

4.66.2.11 `~__versa_string()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~~__versa_string () [inline], [noexcept]
```

Destroy the string instance.

Definition at line 250 of file `vstring.h`.

## 4.66.3 Member Function Documentation

4.66.3.1 `append()` [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Append a string to this string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 693 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator+=()`.

**4.66.3.2 append()** [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
 const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n) [inline]
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 710 of file vstring.h.

**4.66.3.3 append()** [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 const _CharT * __s,
 size_type __n) [inline]
```

Append a C substring.

#### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

#### Returns

Reference to this string.

Definition at line 722 of file `vstring.h`.

#### 4.66.3.4 `append()` [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 const _CharT * __s) [inline]
```

Append a C string.

#### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

#### Returns

Reference to this string.

Definition at line 735 of file `vstring.h`.

#### 4.66.3.5 `append()` [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 size_type __n,
 _CharT __c) [inline]
```

Append multiple characters.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>_↵<br/>_n</code> | The number of characters to append. |
| <code>_↵<br/>_c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `n` copies of `c` to this string.

Definition at line 752 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.6 append()** [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append (
 std::initializer_list<_CharT > __l) [inline]
```

Append an `initializer_list` of characters.

**Parameters**

|                                          |                                                            |
|------------------------------------------|------------------------------------------------------------|
| <code>↵<br/>_↵<br/>↵<br/>_↵<br/>/</code> | The <code>initializer_list</code> of characters to append. |
|------------------------------------------|------------------------------------------------------------|

**Returns**

Reference to this string.

Definition at line 762 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

4.66.3.7 `append()` [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Append a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

## Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

Definition at line 781 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.66.3.8 `assign()` [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Set value to contents of another string.

## Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

## Returns

Reference to this string.

Definition at line 804 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=()`.

**4.66.3.9** `assign()` [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 __versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 820 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap()`.

**4.66.3.10** `assign()` [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos,
 size_type __n) [inline]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.



## Exceptions

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 841 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.66.3.11 `assign()` [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
 const _CharT * __s,
 size_type __n) [inline]
```

Set value to a C substring.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

## Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.


Definition at line 858 of file `vstring.h`.

4.66.3.12 `assign()` [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign (
 const _CharT * __s) [inline]
```

Set value to contents of a C string.

**Parameters**

|                                                                                                                       |                      |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__s</code> | The C string to use. |
|-----------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.



Definition at line 874 of file `vstring.h`.

**4.66.3.13 assign()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 size_type __n,
 _CharT __c) [inline]
```

Set value to multiple characters.

**Parameters**

|                                                                                                                         |                                 |
|-------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__n</code> | Length of the resulting string. |
| <a href="#"></a><br><code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 891 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.66.3.14 `assign()` [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Set value to a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

## Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

Definition at line 910 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.66.3.15 `assign()` [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
 std::initializer_list< _CharT > __l) [inline]
```

Set value to an `initializer_list` of characters.

## Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__l</code> | The <code>initializer_list</code> of characters to assign. |
|------------------|------------------------------------------------------------|

## Returns

Reference to this string.

Definition at line 920 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

#### 4.66.3.16 `at()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the string.

##### Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__↔<br/>_n</code> | The index of the character to access. |
|-------------------------|---------------------------------------|

##### Returns

Read-only (const) reference to the character.

##### Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 578 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

#### 4.66.3.17 `at()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::at (
 size_type __n) [inline]
```

Provides access to the data contained in the string.

## Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__↵<br/>_n</code> | The index of the character to access. |
|-------------------------|---------------------------------------|

## Returns

Read/write reference to the character.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 600 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.66.3.18 `back()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

Definition at line 633 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_↵  
string<_CharT, _Traits, _Alloc, _Base>::size()`.

4.66.3.19 `back()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back () const [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 641 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator[]()`, and `__gnu_cxx::__versa_↵  
string<_CharT, _Traits, _Alloc, _Base>::size()`.

**4.66.3.20 begin()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 316 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend()`, and `__gnu_cxx::__versa_↵string< _CharT, _Traits, _Alloc, _Base >::rend()`.

**4.66.3.21 begin()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 327 of file vstring.h.

**4.66.3.22 c\_str()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1648 of file vstring.h.

**4.66.3.23 capacity()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 487 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`, and `__gnu_cxx::__versa_↵string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit()`.

#### 4.66.3.24 `cbegin()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const
[inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 391 of file `vstring.h`.

#### 4.66.3.25 `cend()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 399 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

#### 4.66.3.26 `clear()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear () [inline], [noexcept]
```

Erases the string, making it empty.

Definition at line 515 of file `vstring.h`.

#### 4.66.3.27 `compare()` [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const [inline]
```

Compare to a string.

## Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 2074 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::operator<()`, `__gnu_cxx::operator<=()`, `__gnu_cxx::operator==()`, `__gnu_cxx::operator>()`, and `__gnu_cxx::operator>=()`.

4.66.3.28 `compare()` [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n,
 const __versa_string<_CharT, _Traits, _Alloc, _Base > & __str) const
```

Compare substring to a string.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.



Definition at line 460 of file vstring.tcc.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `std::min()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

#### 4.66.3.29 compare() [3/6]

```
template<typename _CharT , typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos1,
 size_type __n1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos2,
 size_type __n2) const
```

Compare substring to a substring.

##### Parameters

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

##### Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 477 of file vstring.tcc.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `std::min()`.

#### 4.66.3.30 compare() [4/6]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 const _CharT * __s) const
```

Compare to a C string.

## Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|

## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__s`, 0 if their values are equivalent, or > 0 if this string is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and the length of a string constructed from `__s`. The function then compares the two strings by calling `traits::compare(data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 496 of file `vstring.tcc`.

4.66.3.31 `compare()` [5/6]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const
```

Compare substring to a C string.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__s`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and the length of a string constructed from `__s`. The function then compares the two string by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 512 of file `vstring.tcc`.

4.66.3.32 `compare()` [6/6]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const
```

Compare substring against a character array.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(), __s, rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `__s` must have at least `n2` characters, `\0` has no special meaning.

Definition at line 529 of file `vstring.tcc`.

4.66.3.33 `copy()`

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const
```

Copy substring into C string.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

**Returns**

Number of characters actually copied

**Exceptions**

|                                       |                                   |
|---------------------------------------|-----------------------------------|
| <code><i>std::out_of_range</i></code> | If <code>pos &gt; size()</code> . |
|---------------------------------------|-----------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 255 of file `vstring.tcc`.

**4.66.3.34 `crbegin()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 408 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

**4.66.3.35 `crend()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 417 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`.

**4.66.3.36** `data()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT* __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data () const [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 1658 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_↵not_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of()`, and `std::operator<<()`.

**4.66.3.37** `empty()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty () const [inline],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 523 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.38** `end()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 335 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin()`, and `__gnu_cxx::__versa_↵string< _CharT, _Traits, _Alloc, _Base >::rbegin()`.

**4.66.3.39** `end()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 346 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.40** `erase()` [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
 size_type __pos = 0,
 size_type __n = npos) [inline]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |
|--------------------------------|---------------------------------------------------------|

Removes `__n` characters from this string starting at `__pos`. The length of the string is reduced by `__n`. If there are  $< __n$  characters to remove, the remainder of the string is truncated. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1174 of file `vstring.h`.

**4.66.3.41** `erase()` [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
 const_iterator __position) [inline]
```

Remove one character.

#### Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

#### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1191 of file `vstring.h`.

#### 4.66.3.42 `erase()` [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Remove a range of characters.

#### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

#### Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 1216 of file `vstring.h`.

**4.66.3.43** `find()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a C substring.

**Parameters**

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                       |
| <code>__pos</code> | Index of character to search from.                        |
| <code>__n</code>   | Number of characters from <code>__s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 270 of file `vstring.tcc`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of()`.

**4.66.3.44** `find()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |



**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1694 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.45 `find()`** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1709 of file `vstring.h`.

**4.66.3.46 `find()`** [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 294 of file `vstring.tcc`.

**4.66.3.47 find\_first\_not\_of() [1/4]**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1926 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

4.66.3.48 `find_first_not_of()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a character not in C substring.

## Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid. |
| <code>__pos</code> | Index of character to search from.       |
| <code>__n</code>   | Number of characters from s to consider. |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 392 of file `vstring.tcc`.

4.66.3.49 `find_first_not_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a character not in C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1957 of file vstring.h.

#### 4.66.3.50 find\_first\_not\_of() [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a different character.

##### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 405 of file vstring.tcc.

#### 4.66.3.51 find\_first\_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

##### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1799 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.52 `find_first_of()`** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find position of a character of C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 353 of file `vstring.tcc`.

**4.66.3.53 `find_first_of()`** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline]
```

Find position of a character of C string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1829 of file `vstring.h`.

**4.66.3.54 find\_first\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(c, pos)`.

Definition at line 1848 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find()`.

4.66.3.55 `find_last_not_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1989 of file `vstring.h`.

4.66.3.56 `find_last_not_of()` [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a character not in C substring.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 417 of file vstring.tcc.

#### 4.66.3.57 find\_last\_not\_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline]
```

Find last position of a character not in C string.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

##### Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 2020 of file vstring.h.

#### 4.66.3.58 find\_last\_not\_of() [4/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |



**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 439 of file `vstring.tcc`.

**4.66.3.59 `find_last_of()`** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1863 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.60 `find_last_of()`** [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a character of C substring.

**Parameters**

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.  |
| <code>__pos</code> | Index of character to search back from.    |
| <code>__n</code>   | Number of characters from s to search for. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 370 of file `vstring.tcc`.

**4.66.3.61 `find_last_of()`** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1893 of file `vstring.h`.

**4.66.3.62 `find_last_of()`** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(c, pos)`.

Definition at line 1912 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::rfind()`.

4.66.3.63 `front()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 617 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

4.66.3.64 `front()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::front () const
[inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 625 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator[]()`.

#### 4.66.3.65 get\_allocator()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ()
const [inline], [noexcept]
```

Return copy of allocator used to construct this string.

Definition at line 1665 of file vstring.h.

#### 4.66.3.66 insert() [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

##### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                              |
| <code>__c</code> | The character to insert.                                    |

##### Returns

Iterator referencing the first inserted char.

##### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 941 of file vstring.h.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`.

4.66.3.67 `insert()` [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline]
```

Insert a range of characters.

## Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__p</code>   | Const_iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                             |
| <code>__end</code> | End of range.                                               |

## Returns

Iterator referencing the first inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[beg,end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 985 of file `vstring.h`.

4.66.3.68 `insert()` [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 std::initializer_list< _CharT > __l) [inline]
```

Insert an `initializer_list` of characters.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__p</code> | Const_iterator referencing location in string to insert at. |
| <code>__l</code> | The <code>initializer_list</code> of characters to insert.  |

**Returns**

Iterator referencing the first inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Definition at line 1021 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert()`.

**4.66.3.69 insert()** [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Insert value of a string.

**Parameters**

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1038 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

4.66.3.70 `insert()` [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos2,
 size_type __n) [inline]
```

Insert a substring.

## Parameters

|                     |                                                       |
|---------------------|-------------------------------------------------------|
| <code>__pos1</code> | Iterator referencing location in string to insert at. |
| <code>__str</code>  | The string to insert.                                 |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert.    |
| <code>__n</code>    | Number of characters to insert.                       |

## Returns

Reference to this string.

## Exceptions

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |

Starting at `__pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1061 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

4.66.3.71 `insert()` [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n) [inline]
```

Insert a C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |
| <code>__n</code>   | The number of characters to insert.                   |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1084 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**4.66.3.72 insert()** [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 const _CharT * __s) [inline]
```

Insert a C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__pos</code> | Iterator referencing location in string to insert at. |
| <code>__s</code>   | The C string to insert.                               |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |



Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1103 of file `vstring.h`.

#### 4.66.3.73 `insert()` [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

##### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1127 of file `vstring.h`.

#### 4.66.3.74 `insert()` [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::insert (
 const_iterator __p,
 _CharT __c) [inline]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1146 of file `vstring.h`.

**4.66.3.75 `length()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 432 of file `vstring.h`.

**4.66.3.76 `max_size()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const [inline],
[noexcept]
```

Returns the `size()` of the largest possible string.

Definition at line 437 of file `vstring.h`.

Referenced by `std::getline()`.

4.66.3.77 `operator+=()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::operator+= (
 const __versa_string<_CharT, _Traits, _Alloc, _Base> & __str) [inline]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 652 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

**4.66.3.78 operator+=()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

Definition at line 661 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::append()`.

**4.66.3.79 operator+=()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 _CharT __c) [inline]
```

Append a character.

## Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|

## Returns

Reference to this string.

Definition at line 670 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back()`.

4.66.3.80 `operator+=()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
 std::initializer_list< _CharT > __l) [inline]
```

Append an `initializer_list` of characters.

## Parameters

|                  |                                                                 |
|------------------|-----------------------------------------------------------------|
| <code>__l</code> | The <code>initializer_list</code> of characters to be appended. |
|------------------|-----------------------------------------------------------------|

## Returns

Reference to this string.

Definition at line 683 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`.

4.66.3.81 `operator=()` [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Assign the value of `str` to this string.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

Definition at line 257 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`.

**4.66.3.82 operator=()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 __versa_string< _CharT, _Traits, _Alloc, _Base > && __str) [inline], [noexcept]
```

String move assignment operator.

**Parameters**

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The contents of `__str` are moved into this string (without copying). `__str` is a valid, but unspecified string.

Definition at line 269 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::swap()`.

**4.66.3.83 operator=()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 std::initializer_list< _CharT > __l) [inline]
```

Set value to string constructed from initializer list.

**Parameters**

|                |                                     |
|----------------|-------------------------------------|
| <code>↩</code> | <code>std::initializer_list.</code> |
| <code>↩</code> |                                     |
| <code>↩</code> |                                     |
| <code>↩</code> |                                     |
| <code>/</code> |                                     |

Definition at line 281 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

#### 4.66.3.84 `operator=()` [ 4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 const _CharT * __s) [inline]
```

Copy contents of `__s` into this string.

##### Parameters

|                  |                                |
|------------------|--------------------------------|
| <code>__s</code> | Source null-terminated string. |
|------------------|--------------------------------|

Definition at line 293 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

#### 4.66.3.85 `operator=()` [ 5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
 _CharT __c) [inline]
```

Set value to string of length 1.

##### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__c</code> | Source character. |
|------------------|-------------------|

Assigning to a character makes this string length 1 and `(*this)[0] == __c`.

Definition at line 304 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`.

**4.66.3.86** `operator[]()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
 size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 538 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front()`.

**4.66.3.87** `operator[]()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
 size_type __pos) [inline], [noexcept]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 555 of file `vstring.h`.



**4.66.3.88** `pop_back()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

Definition at line 1236 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.89** `push_back()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (
 _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to append. |
|------------------|----------------------|

Definition at line 789 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::operator+()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+=()`.

**4.66.3.90** `rbegin()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 355 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

**4.66.3.91** `rbegin()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 364 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end()`.

**4.66.3.92** `rend()` [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 373 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`.

**4.66.3.93** `rend()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 382 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin()`.

**4.66.3.94** `replace()` [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]
```

Replace characters with value from another string.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |

Removes the characters in the range `[pos,pos+n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1258 of file `vstring.h`.

References `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size()`.

Referenced by `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert()`, and `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace()`.

4.66.3.95 `replace()` [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos1,
 size_type __n1,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos2,
 size_type __n2) [inline]
```

Replace characters with value from another string.

## Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos1</code> | Index of first character to replace.    |
| <code>__n1</code>   | Number of characters to be replaced.    |
| <code>__str</code>  | String to insert.                       |
| <code>__pos2</code> | Index of first character of str to use. |
| <code>__n2</code>   | Number of characters from str to use.   |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                          |
|--------------------------|----------------------------------------------------------|
| <i>std::out_of_range</i> | If <i>__pos1</i> > size() or <i>__pos2</i> > str.size(). |
| <i>std::length_error</i> | If new length exceeds max_size().                        |

Removes the characters in the range [pos1,pos1 + n) from this string. In place, the value of *\_\_str* is inserted. If *\_\_pos* is beyond end of string, out\_of\_range is thrown. If the length of the result exceeds max\_size(), length\_error is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1281 of file vstring.h.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`.

**4.66.3.96 replace()** [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) [inline]
```

Replace characters with value of a C substring.

**Parameters**

|              |                                              |
|--------------|----------------------------------------------|
| <i>__pos</i> | Index of first character to replace.         |
| <i>__n1</i>  | Number of characters to be replaced.         |
| <i>__s</i>   | C string to insert.                          |
| <i>__n2</i>  | Number of characters from <i>__s</i> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                   |
|--------------------------|-----------------------------------|
| <i>std::out_of_range</i> | If <i>__pos1</i> > size().        |
| <i>std::length_error</i> | If new length exceeds max_size(). |

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the first `__n2` characters of `__s` are inserted, or all of `__s` if `__n2` is too large. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1309 of file `vstring.h`.

#### 4.66.3.97 `replace()` [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline]
```

Replace characters with value of a C string.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos,pos + n1)` from this string. In place, the characters of `__s` are inserted. If `pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown.

The value of the string doesn't change if an error is thrown.

Definition at line 1333 of file `vstring.h`.

#### 4.66.3.98 `replace()` [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
```

```

size_type __pos,
size_type __n1,
size_type __n2,
_CharT __c) [inline]

```

Replace characters with multiple characters.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1357 of file `vstring.h`.

#### 4.66.3.99 `replace()` [6/11]

```

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) [inline]

```

Replace range of characters with string.

#### Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1376 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::size()`.

**4.66.3.100 `replace()`** [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const _CharT * __s,
 size_type __n) [inline]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.     |
| <code>__i2</code> | Iterator referencing end of range to replace.       |
| <code>__s</code>  | C string value to insert.                           |
| <code>__n</code>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range [i1,i2). In place, the first *n* characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1399 of file `vstring.h`.

#### 4.66.3.101 `replace()` [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 const _CharT * __s) [inline]
```

Replace range of characters with C string.

##### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range [i1,i2). In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1425 of file `vstring.h`.

#### 4.66.3.102 `replace()` [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
```



```
const_iterator __i2,
size_type __n,
_CharT __c) [inline]
```

Replace range of characters with multiple characters.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1450 of file `vstring.h`.

#### 4.66.3.103 `replace()` [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
__versa_string& __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline]
```

Replace range of characters with range.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1479 of file `vstring.h`.

**4.66.3.104 replace()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string& __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
 const_iterator __i1,
 const_iterator __i2,
 std::initializer_list< _CharT > __l) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[i1,i2)`. In place, characters in the range `[k1,k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 1583 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`.

#### 4.66.3.105 `reserve()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::reserve (
 size_type __res_arg = 0) [inline]
```

Attempt to preallocate enough memory for specified number of characters.

##### Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

##### Exceptions

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 508 of file `vstring.h`.

Referenced by `__gnu_cxx::operator+()`.

#### 4.66.3.106 `resize()` [1/2]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::resize (
 size_type __n,
 _CharT __c)
```

Resizes the string to the specified number of characters.

##### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
| <code>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 50 of file `vstring.tcc`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize()`.

#### 4.66.3.107 `resize()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
 size_type __n) [inline]
```

Resizes the string to the specified number of characters.

##### Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__n</code> | Number of characters the string should contain. |
|------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as `char`, this means setting them to 0.

Definition at line 464 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::resize()`.

#### 4.66.3.108 `rfind()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
 const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1739 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::find_last_of()`.

**4.66.3.109 `rfind()`** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const
```

Find last position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 313 of file `vstring.tcc`.

**4.66.3.110 `rfind()`** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline]
```

Find last position of a C string.

**Parameters**

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |

**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1769 of file `vstring.h`.

**4.66.3.111 `rfind()`** [ 4 / 4 ]

```
template<typename _CharT, typename _Traits , typename _Alloc , template< typename, typename,
typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↵
_Traits, _Alloc, _Base >::rfind (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 335 of file `vstring.tcc`.

**4.66.3.112 `shrink_to_fit()`**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
void __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit () [inline],
[noexcept]
```

A non-binding request to reduce capacity() to size().

Definition at line 470 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::capacity()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size()`.

#### 4.66.3.113 `size()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::size () const [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 426 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::append()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::at()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::begin()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::cend()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::compare()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::empty()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::end()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_not_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_first_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::find_last_of()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::insert()`, `__gnu_cxx::operator+()`, `std::operator<<()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::pop_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::push_back()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::replace()`, and `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::shrink_to_fit()`.

#### 4.66.3.114 `substr()`

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base>::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline]
```

Get a substring.

##### Parameters

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                |                                   |
|--------------------------------|-----------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> . |
|--------------------------------|-----------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 2053 of file `vstring.h`.

References `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::__versa_string()`.

**4.66.3.115 swap()**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (
 __versa_string< _CharT, _Traits, _Alloc, _Base > & __s) [inline], [noexcept]
```

Swap contents with another string.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__↔</code> | String to swap with. |
| <code>__s</code> |                      |

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 1637 of file `vstring.h`.

Referenced by `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::assign()`, `__gnu_cxx::__versa_string<_CharT, _Traits, _Alloc, _Base >::operator=()`, and `__gnu_cxx::swap()`.

**4.66.4 Member Data Documentation****4.66.4.1 npos**

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< __↔
CharT, _Traits, _Alloc, _Base >::npos [static]
```



Value returned by various member functions when they fail.

Definition at line 82 of file `vstring.h`.

The documentation for this class was generated from the following files:

- [vstring.h](#)
- [vstring.tcc](#)

## 4.67 `__gnu_debug::_After_nth_from<_Iterator>` Class Template Reference

### Public Member Functions

- `_After_nth_from` (const difference\_type &\_\_n, const \_Iterator &\_\_base)
- `bool operator()` (const \_Iterator &\_\_x) const

#### 4.67.1 Detailed Description

```
template<typename _Iterator>
class __gnu_debug::_After_nth_from<_Iterator>
```

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.

Definition at line 74 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

## 4.68 `std::_Base_bitset<_Nw>` Struct Template Reference

### Public Types

- `typedef unsigned long _WordT`

## Public Member Functions

- constexpr **\_Base\_bitset** (unsigned long long \_\_val) noexcept
- template<size\_t \_Nb>  
bool **\_M\_are\_all** () const noexcept
- void **\_M\_do\_and** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- size\_t **\_M\_do\_count** () const noexcept
- size\_t **\_M\_do\_find\_first** (size\_t) const noexcept
- size\_t **\_M\_do\_find\_next** (size\_t, size\_t) const noexcept
- void **\_M\_do\_flip** () noexcept
- void **\_M\_do\_left\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_or** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- void **\_M\_do\_reset** () noexcept
- void **\_M\_do\_right\_shift** (size\_t \_\_shift) noexcept
- void **\_M\_do\_set** () noexcept
- unsigned long long **\_M\_do\_to\_ullong** () const
- unsigned long **\_M\_do\_to\_ulong** () const
- void **\_M\_do\_xor** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) noexcept
- const \_WordT \* **\_M\_getdata** () const noexcept
- \_WordT & **\_M\_getword** (size\_t \_\_pos) noexcept
- constexpr \_WordT **\_M\_getword** (size\_t \_\_pos) const noexcept
- \_WordT & **\_M\_hiword** () noexcept
- constexpr \_WordT **\_M\_hiword** () const noexcept
- bool **\_M\_is\_any** () const noexcept
- bool **\_M\_is\_equal** (const [\\_Base\\_bitset](#)< \_Nw > &\_\_x) const noexcept

## Static Public Member Functions

- static constexpr \_WordT **\_S\_maskbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbit** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichbyte** (size\_t \_\_pos) noexcept
- static constexpr size\_t **\_S\_whichword** (size\_t \_\_pos) noexcept

## Public Attributes

- \_WordT [\\_M\\_w](#) [\_Nw]

### 4.68.1 Detailed Description

```
template<size_t _Nw>
struct std::_Base_bitset< _Nw >
```

Base class, general case. It is a class invariant that \_Nw will be nonnegative.

See documentation for bitset.

Definition at line 75 of file bitset.

## 4.68.2 Member Data Documentation

4.68.2.1 `_M_w`

```
template<size_t _Nw>
_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
```

0 is the least significant word.

Definition at line 80 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

4.69 `std::_Base_bitset< 0 >` Struct Template Reference

## Public Types

- typedef unsigned long `_WordT`

## Public Member Functions

- constexpr `_Base_bitset` (unsigned long long) noexcept
- template<size\_t \_Nb>  
bool `_M_are_all` () const noexcept
- void `_M_do_and` (const `_Base_bitset< 0 >` &) noexcept
- size\_t `_M_do_count` () const noexcept
- size\_t `_M_do_find_first` (size\_t) const noexcept
- size\_t `_M_do_find_next` (size\_t, size\_t) const noexcept
- void `_M_do_flip` () noexcept
- void `_M_do_left_shift` (size\_t) noexcept
- void `_M_do_or` (const `_Base_bitset< 0 >` &) noexcept
- void `_M_do_reset` () noexcept
- void `_M_do_right_shift` (size\_t) noexcept
- void `_M_do_set` () noexcept
- unsigned long long `_M_do_to_ullong` () const noexcept
- unsigned long `_M_do_to_ulong` () const noexcept
- void `_M_do_xor` (const `_Base_bitset< 0 >` &) noexcept
- `_WordT` & `_M_getword` (size\_t) noexcept
- constexpr `_WordT` `_M_getword` (size\_t) const noexcept
- constexpr `_WordT` `_M_hiword` () const noexcept
- bool `_M_is_any` () const noexcept
- bool `_M_is_equal` (const `_Base_bitset< 0 >` &) const noexcept

### Static Public Member Functions

- static constexpr `_WordT` **`_S_maskbit`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichbit`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichbyte`** (`size_t __pos`) noexcept
- static constexpr `size_t` **`_S_whichword`** (`size_t __pos`) noexcept

#### 4.69.1 Detailed Description

```
template<>
struct std::_Base_bitset< 0 >
```

Base class, specialization for no storage (zero-length bitset).

See documentation for `bitset`.

Definition at line 523 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

#### 4.70 `std::_Base_bitset< 1 >` Struct Template Reference

##### Public Types

- typedef unsigned long **`_WordT`**

##### Public Member Functions

- constexpr **`_Base_bitset`** (`unsigned long long __val`) noexcept
- template<`size_t _Nb`>  
  bool **`_M_are_all`** () const noexcept
- void **`_M_do_and`** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- `size_t` **`_M_do_count`** () const noexcept
- `size_t` **`_M_do_find_first`** (`size_t __not_found`) const noexcept
- `size_t` **`_M_do_find_next`** (`size_t __prev`, `size_t __not_found`) const noexcept
- void **`_M_do_flip`** () noexcept
- void **`_M_do_left_shift`** (`size_t __shift`) noexcept
- void **`_M_do_or`** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- void **`_M_do_reset`** () noexcept
- void **`_M_do_right_shift`** (`size_t __shift`) noexcept
- void **`_M_do_set`** () noexcept
- unsigned long long **`_M_do_to_ullong`** () const noexcept
- unsigned long **`_M_do_to_ulong`** () const noexcept
- void **`_M_do_xor`** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) noexcept
- const `_WordT` \* **`_M_getdata`** () const noexcept
- `_WordT` & **`_M_getword`** (`size_t`) noexcept
- constexpr `_WordT` **`_M_getword`** (`size_t`) const noexcept
- `_WordT` & **`_M_hiword`** () noexcept
- constexpr `_WordT` **`_M_hiword`** () const noexcept
- bool **`_M_is_any`** () const noexcept
- bool **`_M_is_equal`** (const [\\_Base\\_bitset< 1 >](#) &\_\_x) const noexcept

## Static Public Member Functions

- static constexpr `_WordT _S_maskbit` (`size_t __pos`) noexcept
- static constexpr `size_t _S_whichbit` (`size_t __pos`) noexcept
- static constexpr `size_t _S_whichbyte` (`size_t __pos`) noexcept
- static constexpr `size_t _S_whichword` (`size_t __pos`) noexcept

## Public Attributes

- `_WordT _M_w`

## 4.70.1 Detailed Description

```
template<>
struct std::_Base_bitset< 1 >
```

Base class, specialization for a single word.

See documentation for `bitset`.

Definition at line 376 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

4.71 `__gnu_debug::_BeforeBeginHelper< _Sequence >` Struct Template Reference

## Static Public Member Functions

- template<typename `_Iterator` , typename `_Category` >  
static bool `_S_Is` (const [\\_Safe\\_iterator](#)< `_Iterator`, `_Sequence`, `_Category` > &)
- template<typename `_Iterator` , typename `_Category` >  
static bool `_S_Is_Beginnest` (const [\\_Safe\\_iterator](#)< `_Iterator`, `_Sequence`, `_Category` > &\_\_it)

## 4.71.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper< _Sequence >
```

Helper struct to deal with sequence offering a `before_begin` iterator.

Definition at line 70 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 4.72 `std::_Bind<_Signature >` Struct Template Reference

### 4.72.1 Detailed Description

```
template<typename _Signature>
struct std::_Bind<_Signature >
```

Type of the function object returned from `bind()`.

Definition at line 398 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.73 `std::_Bind_result<_Result, _Signature >` Struct Template Reference

### 4.73.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::_Bind_result<_Result, _Signature >
```

Type of the function object returned from `bind<R>()`.

Definition at line 549 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.74 `__gnu_cxx::__detail::_Bitmap_counter<_Tp >` Class Template Reference

### Public Member Functions

- `_Bitmap_counter` ([\\_BPVector](#) &Rvbp, long \_\_index=-1)
- pointer `_M_base` () const throw ()
- bool `_M_finished` () const throw ()
- `std::size_t * _M_get` () const throw ()
- `_Index_type _M_offset` () const throw ()
- void `_M_reset` (long \_\_index=-1) throw ()
- void `_M_set_internal_bitmap` (`std::size_t * __new_internal_marker`) throw ()
- `_Index_type _M_where` () const throw ()
- [\\_Bitmap\\_counter](#) & `operator++` () throw ()

## 4.74.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter<_Tp >
```

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.

Definition at line 395 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

4.75 `std::__detail::_BracketMatcher<_TraitsT, __icase, __collate >` Struct Template Reference

## Public Types

- typedef `_TraitsT::char_class_type` **\_CharClassT**
- typedef `_TransT::_CharT` **\_CharT**
- typedef `_TraitsT::string_type` **\_StringT**
- typedef `_TransT::_StrTransT` **\_StrTransT**
- typedef `_RegexTranslator<_TraitsT, __icase, __collate >` **\_TransT**

## Public Member Functions

- **\_BracketMatcher** (bool `__is_non_matching`, const `_TraitsT &__traits`)
- void **\_M\_add\_char** (`_CharT __c`)
- void **\_M\_add\_character\_class** (const `_StringT &__s`, bool `__neg`)
- `_StringT` **\_M\_add\_collate\_element** (const `_StringT &__s`)
- void **\_M\_add\_equivalence\_class** (const `_StringT &__s`)
- void **\_M\_make\_range** (`_CharT __l`, `_CharT __r`)
- void **\_M\_ready** ()
- bool **operator()** (`_CharT __ch`) const

## 4.75.1 Detailed Description

```
template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher<_TraitsT, __icase, __collate >
```

Matches a character range (bracket expression)

Definition at line 49 of file `regex_compiler.h`.

The documentation for this struct was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

## 4.76 `__gnu_cxx::_Caster<_ToType>` Struct Template Reference

### Public Types

- `typedef _ToType::element_type * type`

#### 4.76.1 Detailed Description

```
template<typename _ToType>
struct __gnu_cxx::_Caster<_ToType>
```

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef `element_type` which names the type it points to. 2) has a `get()` const method which returns `element_type*`. 3) has a constructor which can take one `element_type*` argument. This type supports the semantics of the pointer cast operators (below.)

Definition at line 52 of file `cast.h`.

The documentation for this struct was generated from the following file:

- [cast.h](#)

## 4.77 `__gnu_cxx::_Char_types<_CharT>` Struct Template Reference

### Public Types

- `typedef unsigned long int_type`
- `typedef std::streamoff off_type`
- `typedef std::streampos pos_type`
- `typedef std::mbstate\_t state_type`

#### 4.77.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::_Char_types<_CharT>
```

Mapping from character type to associated types.

### Note

This is an implementation class for the generic version of `char_traits`. It defines `int_type`, `off_type`, `pos_type`, and `state_type`. By default these are unsigned long, `streamoff`, `streampos`, and `mbstate_t`. Users who need a different set of types, but who don't need to change the definitions of any function defined in `char_traits`, can specialize `__gnu_cxx::_Char_types` while leaving `__gnu_cxx::char_traits` alone.

Definition at line 65 of file `char_traits.h`.

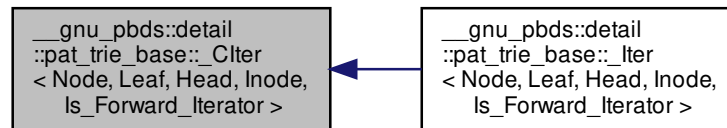
The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)



#### 4.78 `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Clter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:



##### Public Types

- typedef allocator\_type **\_Alloc**
- typedef Node::allocator\_type **allocator\_type**
- typedef type\_traits::const\_pointer **const\_pointer**
- typedef type\_traits::const\_reference **const\_reference**
- typedef allocator\_type::difference\_type **difference\_type**
- typedef [rebind\\_traits](#)< \_Alloc, Head >::pointer **head\_pointer**
- typedef Inode::iterator **inode\_iterator**
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::pointer **inode\_pointer**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::pointer **leaf\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Node >::pointer **node\_pointer**
- typedef type\_traits::pointer **pointer**
- typedef type\_traits::reference **reference**
- typedef Node::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

##### Public Member Functions

- **\_Clter** (node\_pointer p\_nd=0)
- **\_Clter** (const [\\_Clter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool **operator!=** (const [\\_Clter](#) &other) const
- bool **operator!=** (const [\\_Clter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const
- const\_reference **operator\*** () const
- [\\_Clter](#) & **operator++** ()
- [\\_Clter](#) **operator++** (int)
- [\\_Clter](#) & **operator--** ()
- [\\_Clter](#) **operator--** (int)
- const\_pointer **operator->** () const
- [\\_Clter](#) & **operator=** (const [\\_Clter](#) &other)
- [\\_Clter](#) & **operator=** (const [\\_Clter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other)
- bool **operator==** (const [\\_Clter](#) &other) const
- bool **operator==** (const [\\_Clter](#)< Node, Leaf, Head, Inode, !Is\_Forward\_Iterator > &other) const

#### Public Attributes

- node\_pointer **m\_p\_nd**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### Static Protected Member Functions

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

#### 4.78.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Citer< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Const iterator.

Definition at line 487 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 4.79 std::\_\_detail::\_Compiler< \_TraitsT > Class Template Reference

#### Public Types

- typedef \_TraitsT::char\_type **\_CharT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef const \_CharT \* **\_IterT**
- typedef \_NFA< \_TraitsT > **\_RegexT**

#### Public Member Functions

- **\_Compiler** (\_IterT \_\_b, \_IterT \_\_e, const typename \_TraitsT::locale\_type &\_\_traits, [\\_FlagT](#) \_\_flags)
- [shared\\_ptr](#)< const \_RegexT > **\_M\_get\_nfa** ()

## 4.79.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::_Compiler<_TraitsT >
```

Builds an NFA from an input iterator range.

The `_TraitsT` type should fulfill requirements [28.3].

Definition at line 57 of file `regex_compiler.h`.

The documentation for this class was generated from the following files:

- [regex\\_compiler.h](#)
- [regex\\_compiler.tcc](#)

4.80 `std::__parallel::_CRandNumber<_MustBeInt >` Struct Template Reference

## Public Member Functions

- `int operator() (int __limit)`

## 4.80.1 Detailed Description

```
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber<_MustBeInt >
```

Functor wrapper for `std::rand()`.

Definition at line 1529 of file `algo.h`.

The documentation for this struct was generated from the following file:

- [algo.h](#)

4.81 `std::__detail::_Default_ranged_hash` Struct Reference

## 4.81.1 Detailed Description

Default ranged hash function `H`. In principle it should be a function object composed from objects of type `H1` and `H2` such that  $h(k, N) = h2(h1(k), N)$ , but that would mean making extra copies of `h1` and `h2`. So instead we'll just use a tag to tell class template `hashtable` to do that composition.

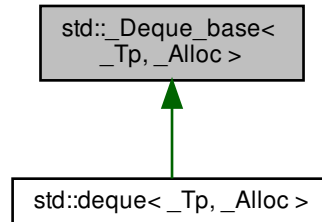
Definition at line 441 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.82 std::Deque\_base< \_Tp, \_Alloc > Class Template Reference

Inheritance diagram for std::Deque\_base< \_Tp, \_Alloc >:



### Protected Types

- enum { **\_S\_initial\_map\_size** }
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Tp\_alloc\_type > **\_Alloc\_traits**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Map\_alloc\_type > **\_Map\_alloc\_traits**
- typedef [\\_Alloc\\_traits::template rebind](#)< \_Ptr >::other **\_Map\_alloc\_type**
- typedef [iterator::\\_Map\\_pointer](#) **\_Map\_pointer**
- typedef [\\_Alloc\\_traits::pointer](#) **\_Ptr**
- typedef [\\_Alloc\\_traits::const\\_pointer](#) **\_Ptr\_const**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Tp >::other **\_Tp\_alloc\_type**
- typedef [\\_Alloc](#) **allocator\_type**
- typedef [\\_Deque\\_iterator](#)< \_Tp, const \_Tp &, \_Ptr\_const > **const\_iterator**
- typedef [\\_Deque\\_iterator](#)< \_Tp, \_Tp &, \_Ptr > **iterator**

### Protected Member Functions

- **\_Deque\_base** (size\_t \_\_num\_elements)
- **\_Deque\_base** (const allocator\_type &\_\_a, size\_t \_\_num\_elements)
- **\_Deque\_base** (const allocator\_type &\_\_a)
- **\_Deque\_base** ([\\_Deque\\_base](#) &&\_\_x)
- **\_Deque\_base** ([\\_Deque\\_base](#) &&\_\_x, const allocator\_type &\_\_a)
- **\_Deque\_base** ([\\_Deque\\_base](#) &&\_\_x, const allocator\_type &\_\_a, size\_t \_\_n)
- [\\_Map\\_pointer](#) **\_M\_allocate\_map** (size\_t \_\_n)
- [\\_Ptr](#) **\_M\_allocate\_node** ()
- void **\_M\_create\_nodes** ([\\_Map\\_pointer](#) \_\_nstart, [\\_Map\\_pointer](#) \_\_nfinish)
- void **\_M\_deallocate\_map** ([\\_Map\\_pointer](#) \_\_p, size\_t \_\_n) noexcept
- void **\_M\_deallocate\_node** ([\\_Ptr](#) \_\_p) noexcept
- void **\_M\_destroy\_nodes** ([\\_Map\\_pointer](#) \_\_nstart, [\\_Map\\_pointer](#) \_\_nfinish) noexcept
- [\\_Map\\_alloc\\_type](#) **\_M\_get\_map\_allocator** () const noexcept
- [\\_Tp\\_alloc\\_type](#) & **\_M\_get\_Tp\_allocator** () noexcept
- const [\\_Tp\\_alloc\\_type](#) & **\_M\_get\_Tp\_allocator** () const noexcept
- void **\_M\_initialize\_map** (size\_t)
- [allocator\\_type](#) **get\_allocator** () const noexcept

### Protected Attributes

- `_Deque_impl _M_impl`

#### 4.82.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_Deque_base<_Tp, _Alloc>
```

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.

Nothing in this class ever constructs or destroys an actual `Tp` element. (Deque handles that itself.) Only/All memory management is performed here.

Definition at line 406 of file `stl_deque.h`.

#### 4.82.2 Member Function Documentation

##### 4.82.2.1 `_M_initialize_map()`

```
template<typename _Tp , typename _Alloc >
void std::_Deque_base<_Tp, _Alloc>::_M_initialize_map (
 size_t __num_elements) [protected]
```

Layout storage.

### Parameters

|                             |                                                        |
|-----------------------------|--------------------------------------------------------|
| <code>__num_elements</code> | The count of T's for which to allocate space at first. |
|-----------------------------|--------------------------------------------------------|

### Returns

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 615 of file `stl_deque.h`.

The documentation for this class was generated from the following file:

- [stl\\_deque.h](#)

### 4.83 `std::_Deque_iterator<_Tp, _Ref, _Ptr>` Struct Template Reference

#### Public Types

- `typedef __ptr_rebind<_Ptr, _Tp> _Elt_pointer`
- `typedef __ptr_rebind<_Ptr, _Elt_pointer> _Map_pointer`
- `typedef _Deque_iterator _Self`
- `typedef __iter<const _Tp> const_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef __iter<_Tp> iterator`
- `typedef std::random_access_iterator_tag iterator_category`
- `typedef _Ptr pointer`
- `typedef _Ref reference`
- `typedef size_t size_type`
- `typedef _Tp value_type`

#### Public Member Functions

- `_Deque_iterator (_Elt_pointer __x, _Map_pointer __y) noexcept`
- `template<typename _Iter, typename = _Require<is_same<_Self, const_iterator>, is_same<_Iter, iterator>>>  
_Deque_iterator (const _Iter &__x) noexcept`
- `_Deque_iterator (const _Deque_iterator &__x) noexcept`
- `iterator _M_const_cast () const noexcept`
- `void _M_set_node (_Map_pointer __new_node) noexcept`
- `reference operator* () const noexcept`
- `_Self & operator++ () noexcept`
- `_Self operator++ (int) noexcept`
- `_Self & operator+= (difference_type __n) noexcept`
- `_Self & operator-- () noexcept`
- `_Self operator-- (int) noexcept`
- `_Self & operator-= (difference_type __n) noexcept`
- `pointer operator-> () const noexcept`
- `_Deque_iterator & operator= (const _Deque_iterator &)=default`
- `reference operator[] (difference_type __n) const noexcept`

#### Static Public Member Functions

- `static size_t _S_buffer_size () noexcept`

#### Public Attributes

- `_Elt_pointer _M_cur`
- `_Elt_pointer _M_first`
- `_Elt_pointer _M_last`
- `_Map_pointer _M_node`

## Friends

- `bool operator!= (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator!= (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `_Self operator+ (const _Self &__x, difference_type __n) noexcept`
- `_Self operator+ (difference_type __n, const _Self &__x) noexcept`
- `difference_type operator- (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
difference_type operator- (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `_Self operator- (const _Self &__x, difference_type __n) noexcept`
- `bool operator< (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator< (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator<= (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator<= (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator== (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator== (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator> (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator> (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`
- `bool operator>= (const _Self &__x, const _Self &__y) noexcept`
- `template<typename _RefR, typename _PtrR >  
bool operator>= (const _Self &__x, const Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept`

## 4.83.1 Detailed Description

```
template<typename _Tp, typename _Ref, typename _Ptr>
struct std::Deque_iterator< _Tp, _Ref, _Ptr >
```

A deque::iterator.

Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.

All the functions are op overloads except for `_M_set_node`.

Definition at line 478 of file `stl_algobase.h`.

## 4.83.2 Member Function Documentation

#### 4.83.2.1 `_M_set_node()`

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
 _Map_pointer __new_node) [inline], [noexcept]
```

Prepares to traverse `new_node`. Sets everything except `_M_cur`, which should therefore be set by the caller immediately afterwards, based on `_M_first` and `_M_last`.

Definition at line 260 of file `stl_deque.h`.

The documentation for this struct was generated from the following files:

- [stl\\_algobase.h](#)
- [stl\\_deque.h](#)

### 4.84 `__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>` Struct Template Reference

#### Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef `std::iterator_traits<_RAIter>` **`_TraitsType`**
- typedef `_TraitsType::value_type` **`_ValueType`**

#### Public Member Functions

- [`\_DRandomShufflingGlobalData`](#) (`_RAIter` & `__source`)

#### Public Attributes

- `_ThreadIndex` \* `_M_bin_proc`
- `_DifferenceType` \*\* `_M_dist`
- `int` `_M_num_bins`
- `int` `_M_num_bits`
- `_RAIter` & `_M_source`
- `_DifferenceType` \* `_M_starts`
- `_ValueType` \*\* `_M_temporaries`

#### 4.84.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>
```

Data known to every thread participating in `__gnu_parallel::__parallel_random_shuffle()`.

Definition at line 52 of file `random_shuffle.h`.



## 4.84.2 Constructor &amp; Destructor Documentation

## 4.84.2.1 \_DRandomShufflingGlobalData()

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_DRandomShufflingGlobalData (
 _RAIter & __source) [inline]
```

Constructor.

Definition at line 83 of file random\_shuffle.h.

## 4.84.3 Member Data Documentation

## 4.84.3.1 \_M\_bin\_proc

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.

Definition at line 74 of file random\_shuffle.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs().

## 4.84.3.2 \_M\_dist

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData<_RAIter>::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.

Dimensions (\_M\_num\_threads + 1) \_\_x (\_M\_num\_bins + 1).

Definition at line 67 of file random\_shuffle.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs(), and \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs↔\_pu().

#### 4.84.3.3 `_M_num_bins`

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins
```

Number of bins to distribute to.

Definition at line 77 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 4.84.3.4 `_M_num_bits`

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.

Definition at line 80 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 4.84.3.5 `_M_source`

```
template<typename _RAIter>
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the `__source`.

Definition at line 59 of file `random_shuffle.h`.

#### 4.84.3.6 `_M_starts`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' `__chunks`.

Definition at line 70 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

## 4.85 \_\_gnu\_parallel::\_DRSSorterPU<\_RAIter, \_RandomNumberGenerator > Struct Template Reference 1063

### 4.84.3.7 \_M\_temporaries

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries
```

Temporary arrays for each thread.

Definition at line 62 of file random\_shuffle.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs().

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 4.85 \_\_gnu\_parallel::\_DRSSorterPU<\_RAIter, \_RandomNumberGenerator > Struct Template Reference

### Public Attributes

- [\\_BinIndex \\_\\_bins\\_end](#)
- [\\_BinIndex \\_M\\_bins\\_begin](#)
- [int \\_M\\_num\\_threads](#)
- [\\_DRandomShufflingGlobalData< \\_RAIter > \\* \\_M\\_sd](#)
- [uint32\\_t \\_M\\_seed](#)

### 4.85.1 Detailed Description

```
template<typename _RAIter, typename _RandomNumberGenerator>
struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >
```

Local data for a thread participating in \_\_gnu\_parallel::\_\_parallel\_random\_shuffle().

Definition at line 91 of file random\_shuffle.h.

### 4.85.2 Member Data Documentation

#### 4.85.2.1 \_\_bins\_end

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end
```

End index for bins taken care of by this thread.

Definition at line 100 of file random\_shuffle.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs().

#### 4.85.2.2 `_M_bins_begin`

```
template<typename _RAIter, typename _RandomNumberGenerator>
__BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin
```

Begin index for bins taken care of by this thread.

Definition at line 97 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`.

#### 4.85.2.3 `_M_num_threads`

```
template<typename _RAIter, typename _RandomNumberGenerator>
int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads
```

Number of threads participating in total.

Definition at line 94 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 4.85.2.4 `_M_sd`

```
template<typename _RAIter, typename _RandomNumberGenerator>
__DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumber↵Generator >::_M_sd
```

Pointer to global data.

Definition at line 106 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

#### 4.85.2.5 `_M_seed`

```
template<typename _RAIter, typename _RandomNumberGenerator>
uint32_t __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed
```

Random `_M_seed` for this thread.

Definition at line 103 of file `random_shuffle.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__parallel_random_shuffle_drs↵_pu()`.

The documentation for this struct was generated from the following file:

- [random\\_shuffle.h](#)

## 4.86 \_\_gnu\_parallel::\_DummyReduct Struct Reference

## Public Member Functions

- `bool operator() (bool, bool) const`

## 4.86.1 Detailed Description

Reduction function doing nothing.

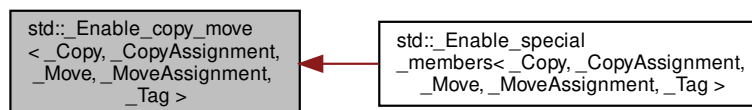
Definition at line 298 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

## 4.87 std::\_Enable\_copy\_move&lt; \_Copy, \_CopyAssignment, \_Move, \_MoveAssignment, \_Tag &gt; Struct Template Reference

Inheritance diagram for `std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



## 4.87.1 Detailed Description

```
template<bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag = void>
struct std::_Enable_copy_move< _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >
```

A mixin helper to conditionally enable or disable the copy/move special members.

## See also

[\\_Enable\\_special\\_members](#)

Definition at line 84 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

## 4.88 `std::_Enable_default_constructor<_Switch, _Tag>` Struct Template Reference

### Public Member Functions

- `constexpr \_Enable\_default\_constructor (\_Enable\_default\_constructor const &) noexcept=default`
- `constexpr \_Enable\_default\_constructor (\_Enable\_default\_constructor &&) noexcept=default`
- `constexpr \_Enable\_default\_constructor (\_Enable\_default\_constructor\_tag)`
- `\_Enable\_default\_constructor & operator= (\_Enable\_default\_constructor const &) noexcept=default`
- `\_Enable\_default\_constructor & operator= (\_Enable\_default\_constructor &&) noexcept=default`

### 4.88.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_default_constructor<_Switch, _Tag>
```

A mixin helper to conditionally enable or disable the default constructor.

See also

[\\_Enable\\_special\\_members](#)

Definition at line 50 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

## 4.89 `std::_Enable_destructor<_Switch, _Tag>` Struct Template Reference

### 4.89.1 Detailed Description

```
template<bool _Switch, typename _Tag = void>
struct std::_Enable_destructor<_Switch, _Tag>
```

A mixin helper to conditionally enable or disable the default destructor.

See also

[\\_Enable\\_special\\_members](#)

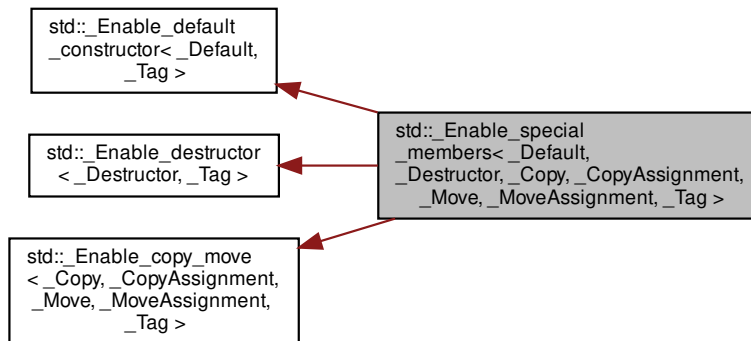
Definition at line 74 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

#### 4.90 `std::Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >` Struct Template Reference

Inheritance diagram for `std::Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >`:



##### 4.90.1 Detailed Description

```
template<bool _Default, bool _Destructor, bool _Copy, bool _CopyAssignment, bool _Move, bool _MoveAssignment, typename _Tag = void>
struct std::Enable_special_members< _Default, _Destructor, _Copy, _CopyAssignment, _Move, _MoveAssignment, _Tag >
```

A mixin helper to conditionally enable or disable the special members.

The `_Tag` type parameter is to make mixin bases unique and thus avoid ambiguities.

Definition at line 97 of file `enable_special_members.h`.

The documentation for this struct was generated from the following file:

- [enable\\_special\\_members.h](#)

#### 4.91 `__gnu_debug::Equal_to< _Type >` Class Template Reference

##### Public Member Functions

- `_Equal_to` (const `_Type` &\_\_v)
- `bool operator()` (const `_Type` &\_\_x) const

#### 4.91.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::_Equal_to< _Type >
```

A simple function object that returns true if the passed-in value is equal to the stored value.

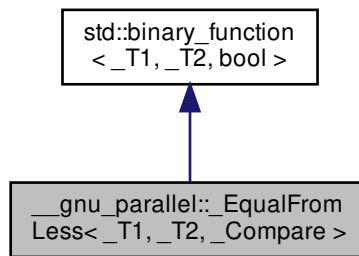
Definition at line 59 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

#### 4.92 \_\_gnu\_parallel::\_EqualFromLess< \_T1, \_T2, \_Compare > Class Template Reference

Inheritance diagram for `__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >`:



##### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

##### Public Member Functions

- **\_EqualFromLess** (`_Compare &__comp`)
- `bool` **operator()** (`const _T1 &__a, const _T2 &__b`)



#### 4.92.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_EqualFromLess<_T1, _T2, _Compare>
```

Constructs predicate for equality from strict weak ordering predicate.

Definition at line 157 of file `base.h`.

#### 4.92.2 Member Typedef Documentation

##### 4.92.2.1 `first_argument_type`

```
typedef _T1 std::binary_function<_T1, _T2, bool>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.92.2.2 `result_type`

```
typedef bool std::binary_function<_T1, _T2, bool>::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.92.2.3 `second_argument_type`

```
typedef _T2 std::binary_function<_T1, _T2, bool>::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

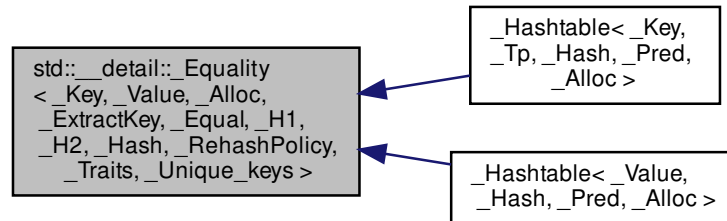
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [base.h](#)

#### 4.93 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >` Struct Template Reference

Inheritance diagram for `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >`:



##### 4.93.1 Detailed Description

```

template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >

```

Primary class template `_Equality`.

This is for implementing equality comparison for unordered containers, per N3068, by John Lakos and Pablo Halpern. Algorithmically, we follow closely the reference implementations therein.

Definition at line 1831 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.94 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

##### Public Types

- using `__hashtable` = `_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Public Member Functions

- `bool _M_equal (const \_\_hashtable &) const`

##### 4.94.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

`unordered_multiset` and `unordered_multimap` specializations.

Definition at line 1890 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.95 `std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

#### Public Types

- using `__hashtable` = `\_Hashtable< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Public Member Functions

- `bool _M_equal (const \_\_hashtable &) const`

##### 4.95.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Equality< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

`unordered_map` and `unordered_set` specializations.

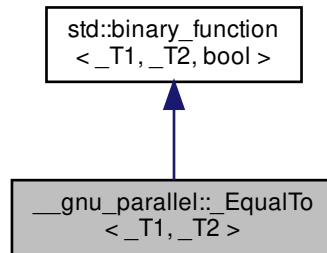
Definition at line 1838 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.96 \_\_gnu\_parallel::\_EqualTo<\_T1, \_T2> Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_EqualTo<\_T1, \_T2>:



##### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

##### Public Member Functions

- `bool` **operator()** (const `_T1` &\_\_t1, const `_T2` &\_\_t2) const

##### 4.96.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo<_T1, _T2>
```

Similar to `std::equal_to`, but allows two different types.

Definition at line 244 of file `base.h`.

##### 4.96.2 Member Typedef Documentation

#### 4.96.2.1 `first_argument_type`

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 4.96.2.2 `result_type`

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 4.96.2.3 `second_argument_type`

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [base.h](#)

## 4.97 `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode>` Class Template Reference

### Public Types

- typedef [iterator\\_traits](#)< \_Bilter >::value\_type `_CharT`
- typedef `_TraitsT::char_class_type` `_ClassT`
- typedef [regex\\_constants::match\\_flag\\_type](#) `_FlagT`
- typedef `_NFA<_TraitsT>` `_NFAT`
- typedef [basic\\_regex](#)< \_CharT, \_TraitsT > `_RegexT`
- typedef `std::vector< sub\_match< _Bilter >, _Alloc >` `_ResultsVec`

### Public Member Functions

- `_Executor` (`_Bilter` \_\_begin, `_Bilter` \_\_end, [\\_ResultsVec](#) &\_\_results, const [\\_RegexT](#) &\_\_re, `_FlagT` \_\_flags)
- `bool` `_M_match` ()
- `bool` `_M_search` ()
- `bool` `_M_search_from_first` ()

## Public Attributes

- `_Bilter _M_begin`
- `_ResultsVec _M_cur_results`
- `_Bilter _M_current`
- `const _Bilter _M_end`
- `_FlagT _M_flags`
- `bool _M_has_sol`
- `const _NFAT & _M_nfa`
- `const _RegexT & _M_re`
- `vector< pair< _Bilter, int > > _M_rep_count`
- `_ResultsVec & _M_results`
- `_State_info< __search_mode, _ResultsVec > _M_states`

### 4.97.1 Detailed Description

```
template<typename _Bilter, typename _Alloc, typename _TraitsT, bool __dfs_mode>
class std::__detail::__Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode >
```

Takes a regex and an input string and does the matching.

The `_Executor` class has two modes: DFS mode and BFS mode, controlled by the template parameter `__dfs_mode`.

Definition at line 59 of file `regex.h`.

The documentation for this class was generated from the following files:

- [regex.h](#)
- [regex\\_executor.h](#)
- [regex\\_executor.tcc](#)

## 4.98 `__gnu_cxx::__ExtPtr_allocator< _Tp >` Class Template Reference

### Public Types

- `typedef _Pointer_adapter< _Relative_pointer_impl< const _Tp > > const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Pointer_adapter< _Relative_pointer_impl< _Tp > > pointer`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `_ExtPtr_allocator` (const `_ExtPtr_allocator` &\_\_rarg) noexcept
- `template<typename _Up >`  
`_ExtPtr_allocator` (const `_ExtPtr_allocator<_Up>` &\_\_rarg) noexcept
- `const std::allocator<_Tp>` & `_M_getUnderlyingImp` () const
- `pointer address` (reference \_\_x) const noexcept
- `const_pointer address` (const\_reference \_\_x) const noexcept
- `pointer allocate` (size\_type \_\_n, const void \*==0)
- `template<typename _Up, typename... _Args>`  
`void construct` (\_Up \*\_\_p, \_Args &&... \_\_args)
- `template<typename... _Args>`  
`void construct` (`pointer` \_\_p, \_Args &&... \_\_args)
- `void deallocate` (`pointer` \_\_p, size\_type \_\_n)
- `template<typename _Up >`  
`void destroy` (\_Up \*\_\_p)
- `void destroy` (`pointer` \_\_p)
- `size_type max_size` () const noexcept
- `template<typename _Up >`  
`bool operator!=` (const `_ExtPtr_allocator<_Up>` &\_\_rarg) const
- `bool operator!=` (const `_ExtPtr_allocator` &\_\_rarg) const
- `template<typename _Up >`  
`bool operator==` (const `_ExtPtr_allocator<_Up>` &\_\_rarg) const
- `bool operator==` (const `_ExtPtr_allocator` &\_\_rarg) const

## Friends

- `template<typename _Up >`  
`void swap` (`_ExtPtr_allocator<_Up>` &, `_ExtPtr_allocator<_Up>` &)

## 4.98.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator<_Tp>
```

An example allocator which uses a non-standard pointer type.

This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See `ext/pointer.h`) Memory allocation in this example is still performed using `std::allocator`.

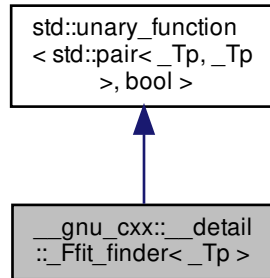
Definition at line 56 of file `extptr_allocator.h`.

The documentation for this class was generated from the following file:

- [extptr\\_allocator.h](#)

#### 4.99 `__gnu_cxx::__detail::_Ffit_finder<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::__detail::_Ffit_finder<_Tp>`:



##### Public Types

- typedef `std::pair<_Tp, _Tp>` `argument_type`
- typedef `bool` `result_type`

##### Public Member Functions

- `std::size_t * _M_get ()` const throw ()
- `_Counter_type _M_offset ()` const throw ()
- `bool operator() (_Block_pair __bp)` throw ()

##### 4.99.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder<_Tp>
```

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.

Definition at line 329 of file `bitmap_allocator.h`.

##### 4.99.2 Member Typedef Documentation



## 4.99.2.1 argument\_type

```
typedef std::pair< _Tp, _Tp > std::unary_function< std::pair< _Tp, _Tp > , bool >::argument_type
[inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 4.99.2.2 result\_type

```
typedef bool std::unary_function< std::pair< _Tp, _Tp > , bool >::result_type [inherited]
```

result\_type is the return type

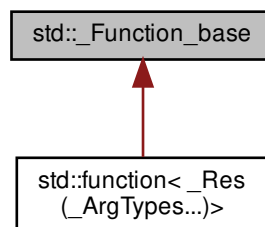
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 4.100 std::\_Function\_base Class Reference

Inheritance diagram for std::\_Function\_base:



## Public Types

- typedef bool(\* **Manager\_type**) (\_Any\_data &, const \_Any\_data &, \_Manager\_operation)

## Public Member Functions

- bool **M\_empty** () const

#### Public Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

#### Static Public Attributes

- `static const size_t _M_max_align`
- `static const size_t _M_max_size`

#### 4.100.1 Detailed Description

Base class of all polymorphic function object wrappers.

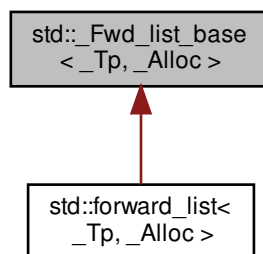
Definition at line 116 of file `std_function.h`.

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

#### 4.101 `std::_Fwd_list_base<_Tp, _Alloc>` Struct Template Reference

Inheritance diagram for `std::_Fwd_list_base<_Tp, _Alloc>`:



#### Public Types

- `typedef \_Fwd\_list\_node<_Tp> _Node`
- `typedef \_Fwd\_list\_const\_iterator<_Tp> const_iterator`
- `typedef \_Fwd\_list\_iterator<_Tp> iterator`

## Public Member Functions

- [\\_Fwd\\_list\\_base](#) ([\\_Node\\_alloc\\_type](#) &&\_\_a)
- [\\_Fwd\\_list\\_base](#) ([\\_Fwd\\_list\\_base](#) &&\_\_lst, [\\_Node\\_alloc\\_type](#) &&\_\_a, [std::true\\_type](#))
- [\\_Fwd\\_list\\_base](#) ([\\_Fwd\\_list\\_base](#) &&\_\_lst, [\\_Node\\_alloc\\_type](#) &&\_\_a)
- [\\_Fwd\\_list\\_base](#) ([\\_Fwd\\_list\\_base](#) &&)=default
- [\\_Node\\_alloc\\_type](#) & [\\_M\\_get\\_Node\\_allocator](#) () noexcept
- const [\\_Node\\_alloc\\_type](#) & [\\_M\\_get\\_Node\\_allocator](#) () const noexcept

## Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< [\\_Node\\_alloc\\_type](#) > [\\_Node\\_alloc\\_traits](#)
- typedef [\\_\\_alloc\\_rebind](#)< [\\_Alloc](#), [\\_Fwd\\_list\\_node](#)< [\\_Tp](#) > > [\\_Node\\_alloc\\_type](#)

## Protected Member Functions

- template<typename... \_Args>  
[\\_Node](#) \* [\\_M\\_create\\_node](#) ([\\_Args](#) &&... \_\_args)
- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_erase\\_after](#) ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_pos)
- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_erase\\_after](#) ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_pos, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_last)
- [\\_Node](#) \* [\\_M\\_get\\_node](#) ()
- template<typename... \_Args>  
[\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_insert\\_after](#) (const\_iterator \_\_pos, [\\_Args](#) &&... \_\_args)
- void [\\_M\\_put\\_node](#) ([\\_Node](#) \* \_\_p)

## Protected Attributes

- [\\_Fwd\\_list\\_impl](#) [\\_M\\_impl](#)

## 4.101.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >
```

Base class for forward\_list.

Definition at line 287 of file forward\_list.h.

The documentation for this struct was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

## 4.102 `std::_Fwd_list_const_iterator<_Tp>` Struct Template Reference

### Public Types

- typedef const `_Fwd_list_node<_Tp>` `_Node`
- typedef `_Fwd_list_const_iterator<_Tp>` `_Self`
- typedef ptrdiff\_t `difference_type`
- typedef `_Fwd_list_iterator<_Tp>` `iterator`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef const \_Tp \* `pointer`
- typedef const \_Tp & `reference`
- typedef \_Tp `value_type`

### Public Member Functions

- `_Fwd_list_const_iterator` (const `_Fwd_list_node_base` \* \_\_n) noexcept
- `_Fwd_list_const_iterator` (const `iterator` & \_\_iter) noexcept
- `_Self _M_next` () const noexcept
- reference `operator*` () const noexcept
- `_Self & operator++` () noexcept
- `_Self operator++` (int) noexcept
- pointer `operator->` () const noexcept

### Public Attributes

- const `_Fwd_list_node_base` \* `_M_node`

### Friends

- bool `operator!=` (const `_Self` & \_\_x, const `_Self` & \_\_y) noexcept
- bool `operator==` (const `_Self` & \_\_x, const `_Self` & \_\_y) noexcept

#### 4.102.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_const_iterator<_Tp>
```

A `forward_list::const_iterator`.

All the functions are op overloads.

Definition at line 210 of file `forward_list.h`.

#### 4.102.2 Friends And Related Function Documentation

## 4.102.2.1 operator!=

```
template<typename _Tp >
bool operator!= (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list const\_iterator inequality comparison.

Definition at line 267 of file forward\_list.h.

## 4.102.2.2 operator==

```
template<typename _Tp >
bool operator== (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list const\_iterator equality comparison.

Definition at line 259 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.103 std::\_Fwd\_list\_iterator&lt;\_Tp&gt; Struct Template Reference

## Public Types

- typedef [\\_Fwd\\_list\\_node](#)<\_Tp> **\_Node**
- typedef [\\_Fwd\\_list\\_iterator](#)<\_Tp> **\_Self**
- typedef ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

## Public Member Functions

- [\\_Fwd\\_list\\_iterator](#) ([\\_Fwd\\_list\\_node\\_base](#) \*\_\_n) noexcept
- [\\_Self](#) **\_M\_next** () const noexcept
- reference **operator\*** () const noexcept
- [\\_Self](#) & **operator++** () noexcept
- [\\_Self](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept

### Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_node](#)

### Friends

- `bool operator!= (const \_Self &__x, const \_Self &__y) noexcept`
- `bool operator== (const \_Self &__x, const \_Self &__y) noexcept`

#### 4.103.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >
```

A forward\_list::iterator.

All the functions are op overloads.

Definition at line 135 of file forward\_list.h.

#### 4.103.2 Friends And Related Function Documentation

##### 4.103.2.1 operator"!=

```
template<typename _Tp >
bool operator!= (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list iterator inequality comparison.

Definition at line 188 of file forward\_list.h.

##### 4.103.2.2 operator==

```
template<typename _Tp >
bool operator== (
 const _Self & __x,
 const _Self & __y) [friend]
```

Forward list iterator equality comparison.

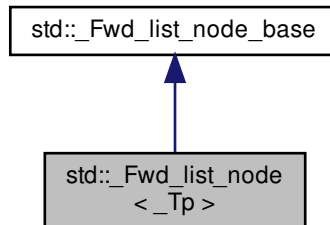
Definition at line 180 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

## 4.104 std::\_Fwd\_list\_node&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::\_Fwd\_list\_node< \_Tp >:



## Public Member Functions

- void **\_M\_reverse\_after** () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_transfer\_after** ([\\_Fwd\\_list\\_node\\_base](#) \* \_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \* \_\_end) noexcept
- [\\_Tp](#) \* **\_M\_valptr** () noexcept
- const [\\_Tp](#) \* **\_M\_valptr** () const noexcept

## Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* **\_M\_next**
- [\\_\\_gnu\\_cxx::\\_\\_aligned\\_buffer< \\_Tp >](#) **\_M\_storage**

## 4.104.1 Detailed Description

```
template<typename _Tp>
struct std::_Fwd_list_node< _Tp >
```

A helper node class for forward\_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.

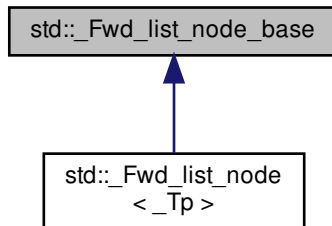
Definition at line 113 of file forward\_list.h.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)

#### 4.105 std::\_Fwd\_list\_node\_base Struct Reference

Inheritance diagram for std::\_Fwd\_list\_node\_base:



##### Public Member Functions

- [\\_Fwd\\_list\\_node\\_base](#) ([\\_Fwd\\_list\\_node\\_base](#) &&\_\_x) noexcept
- [\\_Fwd\\_list\\_node\\_base](#) (const [\\_Fwd\\_list\\_node\\_base](#) &)=delete
- void [\\_M\\_reverse\\_after](#) () noexcept
- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_transfer\\_after](#) ([\\_Fwd\\_list\\_node\\_base](#) \*\_\_begin, [\\_Fwd\\_list\\_node\\_base](#) \*\_\_end) noexcept
- [\\_Fwd\\_list\\_node\\_base](#) & [operator=](#) (const [\\_Fwd\\_list\\_node\\_base](#) &)=delete
- [\\_Fwd\\_list\\_node\\_base](#) & [operator=](#) ([\\_Fwd\\_list\\_node\\_base](#) &&\_\_x) noexcept

##### Public Attributes

- [\\_Fwd\\_list\\_node\\_base](#) \* [\\_M\\_next](#)

##### 4.105.1 Detailed Description

A helper basic node class for `forward_list`. This is just a linked list with nothing inside it. There are purely list shuffling utility methods here.

Definition at line 54 of file `forward_list.h`.

The documentation for this struct was generated from the following file:

- [forward\\_list.h](#)



4.106 `__gnu_parallel::_GuardedIterator<_RAIter, _Compare>` Class Template Reference

## Public Member Functions

- `_GuardedIterator` (`_RAIter __begin`, `_RAIter __end`, `_Compare &__comp`)
- `operator _RAIter` ()
- `std::iterator_traits<_RAIter>::value_type & operator*` ()
- `_GuardedIterator<_RAIter, _Compare> & operator++` ()

## Friends

- `bool operator<` (`_GuardedIterator<_RAIter, _Compare> &__bi1`, `_GuardedIterator<_RAIter, _Compare> &__bi2`)
- `bool operator<=` (`_GuardedIterator<_RAIter, _Compare> &__bi1`, `_GuardedIterator<_RAIter, _Compare> &__bi2`)

## 4.106.1 Detailed Description

```
template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator<_RAIter, _Compare>
```

`_Iterator` wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.

The implicit supremum comes with a performance cost.

Deriving from `_RAIter` is not possible since `_RAIter` need not be a class.

Definition at line 73 of file `multiway_merge.h`.

## 4.106.2 Constructor &amp; Destructor Documentation

4.106.2.1 `_GuardedIterator()`

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator<_RAIter, _Compare>::_GuardedIterator (
 _RAIter __begin,
 _RAIter __end,
 _Compare & __comp) [inline]
```

Constructor. Sets iterator to beginning of sequence.

## Parameters

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| <code>__begin</code> | Begin iterator of sequence.                                      |
| <code>__end</code>   | End iterator of sequence.                                        |
| <code>__comp</code>  | Comparator provided for associated overloaded compare operators. |

Definition at line 91 of file multiway\_merge.h.

#### 4.106.3 Member Function Documentation

##### 4.106.3.1 operator\_RAIter()

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () [inline]
```

Convert to wrapped iterator.

##### Returns

Wrapped iterator.

Definition at line 112 of file multiway\_merge.h.

##### 4.106.3.2 operator\*()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits<_RAIter>::value_type& __gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator* () [inline]
```

Dereference operator.

##### Returns

Referenced element.

Definition at line 107 of file multiway\_merge.h.

##### 4.106.3.3 operator++()

```
template<typename _RAIter, typename _Compare>
__GuardedIterator<_RAIter, _Compare>& __gnu_parallel::_GuardedIterator< _RAIter, _Compare >↵
::operator++ () [inline]
```

Pre-increment operator.

##### Returns

This.

Definition at line 98 of file multiway\_merge.h.

## 4.106.4 Friends And Related Function Documentation

## 4.106.4.1 operator&lt;

```
template<typename _RAIter, typename _Compare>
bool operator< (
 _GuardedIterator< _RAIter, _Compare > & __bi1,
 _GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

## Parameters

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

## Returns

true if less.

Definition at line 120 of file multiway\_merge.h.

## 4.106.4.2 operator&lt;=

```
template<typename _RAIter, typename _Compare>
bool operator<= (
 _GuardedIterator< _RAIter, _Compare > & __bi1,
 _GuardedIterator< _RAIter, _Compare > & __bi2) [friend]
```

Compare two elements referenced by guarded iterators.

## Parameters

|                    |                  |
|--------------------|------------------|
| <code>__bi1</code> | First iterator.  |
| <code>__bi2</code> | Second iterator. |

## Returns

True if less equal.

Definition at line 135 of file multiway\_merge.h.

The documentation for this class was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.107 `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Struct Template Reference

##### 4.107.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_hash_code <←
hash_code>
struct std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Hash_code_base`.

Encapsulates two policy issues that aren't quite orthogonal. (1) the difference between using a ranged hash function and using the combination of a hash function and a range-hashing function. In the former case we don't have such things as hash codes, so we have a dummy type as placeholder. (2) Whether or not we cache hash codes. Caching hash codes is meaningless if we have a ranged hash function.

We also put the key extraction objects here, for convenience. Each specialization derives from one or more of the template parameters to benefit from Ebo. This is important as this type is inherited in some cases by the `_Local_iterator_base` type used to implement `local_iterator` and `const_local_iterator`. As with any iterator type we prefer to make it as small as possible.

Primary template is unused except as a hook for specializations.

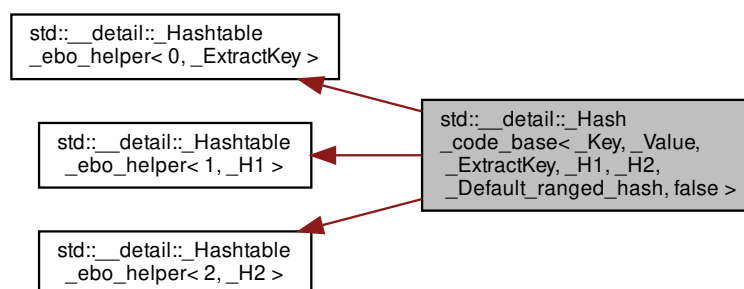
Definition at line 1175 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.108 `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`  
`hash, false >`:



#### Public Types

- `typedef _H1 hasher`

#### Public Member Functions

- `hasher hash_function () const`

#### Protected Types

- `typedef std::size_t __hash_code`
- `typedef \_Hash\_node<_Value, false > __node_type`

#### Protected Member Functions

- `_Hash_code_base (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const \_Default\_ranged\_hash &)`
- `std::size_t _M_bucket_index (const _Key &, __hash_code __c, std::size_t __bkt_count) const`
- `std::size_t _M_bucket_index (const \_\_node\_type * __p, std::size_t __bkt_count) const noexcept(noexcept(declval<const _H1 & >()(declval<const _Key & >())) &&noexcept(declval<const _H2 & >())((__hash_code) 0,(std::size_t) 0)))`
- `void _M_copy_code (\_\_node\_type *, const \_\_node\_type *) const`
- `const _ExtractKey & _M_extract () const`
- `const _H1 & _M_h1 () const`
- `const _H2 & _M_h2 () const`
- `__hash_code _M_hash_code (const _Key &__k) const`
- `void _M_store_code (\_\_node\_type *, __hash_code) const`
- `void _M_swap (\_Hash\_code\_base &__x)`

#### Friends

- `struct \_Local\_iterator\_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >`

#### 4.108.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::__Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, false >
```

Specialization: hash function and range-hashing function, no caching of hash codes. Provides typedef and accessor required by C++ 11.

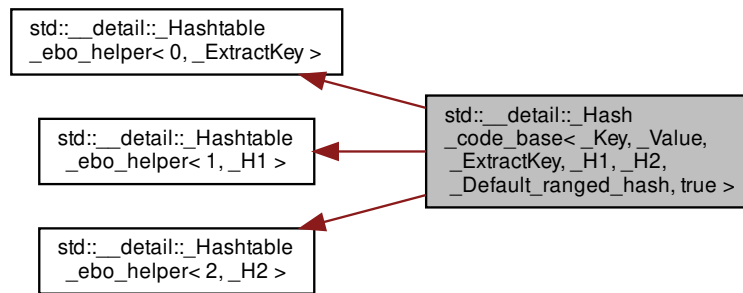
Definition at line 1254 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.109 `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >` Struct Template Reference

Inheritance diagram for `std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`:



##### Public Types

- `typedef _H1 hasher`

##### Public Member Functions

- `hasher hash_function () const`

##### Protected Types

- `typedef std::size_t __hash_code`
- `typedef \_Hash\_node< _Value, true > __node_type`

##### Protected Member Functions

- `__Hash_code_base (const _ExtractKey &__ex, const _H1 &__h1, const _H2 &__h2, const \_Default\_ranged\_hash &)`
- `std::size_t __M_bucket_index (const _Key &, __hash_code __c, std::size_t __bkt_count) const`
- `std::size_t __M_bucket_index (const \_\_node\_type * __p, std::size_t __bkt_count) const noexcept (noexcept (declval< const _H2 & > () ( (__hash_code) 0, (std::size_t) 0)))`
- `void __M_copy_code (\_\_node\_type * __to, const \_\_node\_type * __from) const`
- `const _ExtractKey & __M_extract () const`
- `const _H1 & __M_h1 () const`
- `const _H2 & __M_h2 () const`
- `__hash_code __M_hash_code (const _Key & __k) const`
- `void __M_store_code (\_\_node\_type * __n, __hash_code __c) const`
- `void __M_swap (\_Hash\_code\_base & __x)`

## Friends

- struct `_Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >`

## 4.109.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2>
struct std::__detail::__Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Default_ranged_hash, true >
```

Specialization: hash function and range-hashing function, caching hash codes. H is provided but ignored. Provides typedef and accessor required by C++ 11.

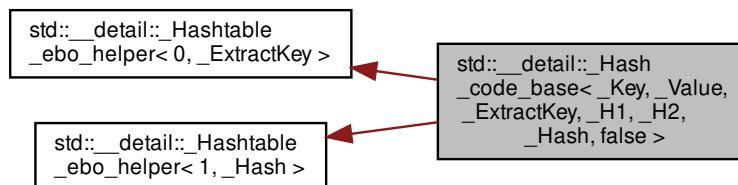
Definition at line 1341 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.110 std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >:



## Protected Types

- typedef void \* `__hash_code`
- typedef `_Hash_node< _Value, false > __node_type`

### Protected Member Functions

- **\_Hash\_code\_base** (const \_ExtractKey & \_\_ex, const \_H1 &, const \_H2 &, const \_Hash & \_\_h)
- **std::size\_t \_M\_bucket\_index** (const \_Key & \_\_k, \_\_hash\_code, std::size\_t \_\_bkt\_count) const
- **std::size\_t \_M\_bucket\_index** (const \_\_node\_type \* \_\_p, std::size\_t \_\_bkt\_count) const noexcept(noexcept(declval<const \_Hash & >()(declval<const \_Key & >()),(std::size\_t) 0)))
- **void \_M\_copy\_code** (\_\_node\_type \*, const \_\_node\_type \*) const
- **const \_ExtractKey & \_M\_extract** () const
- **\_\_hash\_code \_M\_hash\_code** (const \_Key & \_\_key) const
- **const \_Hash & \_M\_ranged\_hash** () const
- **void \_M\_store\_code** (\_\_node\_type \*, \_\_hash\_code) const
- **void \_M\_swap** (\_Hash\_code\_base & \_\_x)

#### 4.110.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::_Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false >
```

Specialization: ranged hash function, no caching hash codes. H1 and H2 are provided but ignored. We define a dummy hash code type.

Definition at line 1181 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 4.111 std::\_\_detail::\_Hash\_node< \_Value, \_Cache\_hash\_code > Struct Template Reference

#### 4.111.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::_Hash_node< _Value, _Cache_hash_code >
```

Primary template struct \_Hash\_node.

Definition at line 256 of file hashtable\_policy.h.

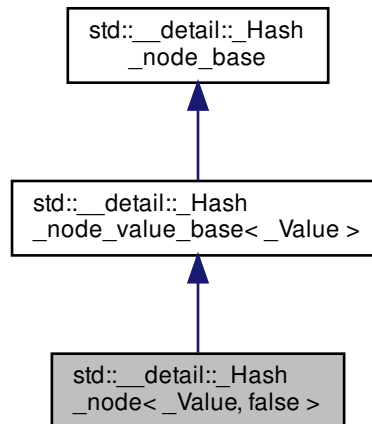
The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



## 4.112 std::\_\_detail::\_Hash\_node&lt; \_Value, false &gt; Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node< \_Value, false >:



## Public Types

- typedef \_Value **value\_type**

## Public Member Functions

- [\\_Hash\\_node](#) \* **\_M\_next** () const noexcept
- \_Value & **\_M\_v** () noexcept
- const \_Value & **\_M\_v** () const noexcept
- \_Value \* **\_M\_valptr** () noexcept
- const \_Value \* **\_M\_valptr** () const noexcept

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* **\_M\_nxt**
- \_\_gnu\_cxx::\_\_aligned\_buffer< \_Value > **\_M\_storage**

#### 4.112.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node< _Value, false >
```

Specialization for nodes without caches, struct `_Hash_node`.

Base class is `__detail::_Hash_node_value_base`.

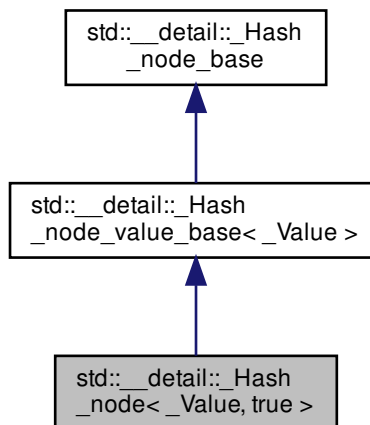
Definition at line 279 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.113 `std::__detail::_Hash_node< _Value, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node< _Value, true >`:



#### Public Types

- typedef `_Value` **value\_type**

#### Public Member Functions

- [\\_Hash\\_node](#) \* **\_M\_next** () const noexcept
- `_Value & _M_v` () noexcept
- const `_Value & _M_v` () const noexcept
- `_Value * _M_valptr` () noexcept
- const `_Value * _M_valptr` () const noexcept

## Public Attributes

- std::size\_t **M\_hash\_code**
- [\\_Hash\\_node\\_base](#) \* **M\_nxt**
- \_\_gnu\_cxx::\_\_aligned\_buffer<\_Value> **M\_storage**

## 4.113.1 Detailed Description

```
template<typename _Value>
struct std::__detail::_Hash_node<_Value, true >
```

Specialization for nodes with caches, struct \_Hash\_node.

Base class is \_\_detail::\_Hash\_node\_value\_base.

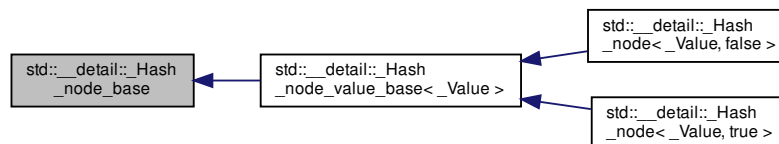
Definition at line 264 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.114 std::\_\_detail::\_Hash\_node\_base Struct Reference

Inheritance diagram for std::\_\_detail::\_Hash\_node\_base:



## Public Member Functions

- **\_Hash\_node\_base** ([\\_Hash\\_node\\_base](#) \*\_\_next) noexcept

## Public Attributes

- [\\_Hash\\_node\\_base](#) \* **M\_nxt**

#### 4.114.1 Detailed Description

struct `_Hash_node_base`

Nodes, used to wrap elements stored in the hash table. A policy template parameter of class template `_Hashtable` controls whether nodes also store a hash code. In some cases (e.g. strings) this may be a performance win.

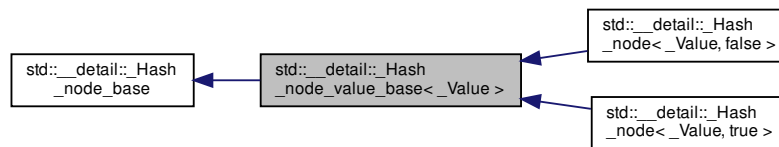
Definition at line 214 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.115 `std::__detail::_Hash_node_value_base<_Value>` Struct Template Reference

Inheritance diagram for `std::__detail::_Hash_node_value_base<_Value>`:



#### Public Types

- typedef `_Value` **value\_type**

#### Public Member Functions

- `_Value & _M_v () noexcept`
- `const _Value & _M_v () const noexcept`
- `_Value * _M_valptr () noexcept`
- `const _Value * _M_valptr () const noexcept`

#### Public Attributes

- `_Hash_node_base * _M_nxt`
- `__gnu_cxx::__aligned_buffer<_Value> _M_storage`

#### 4.115.1 Detailed Description

```
struct Hash node value base
```

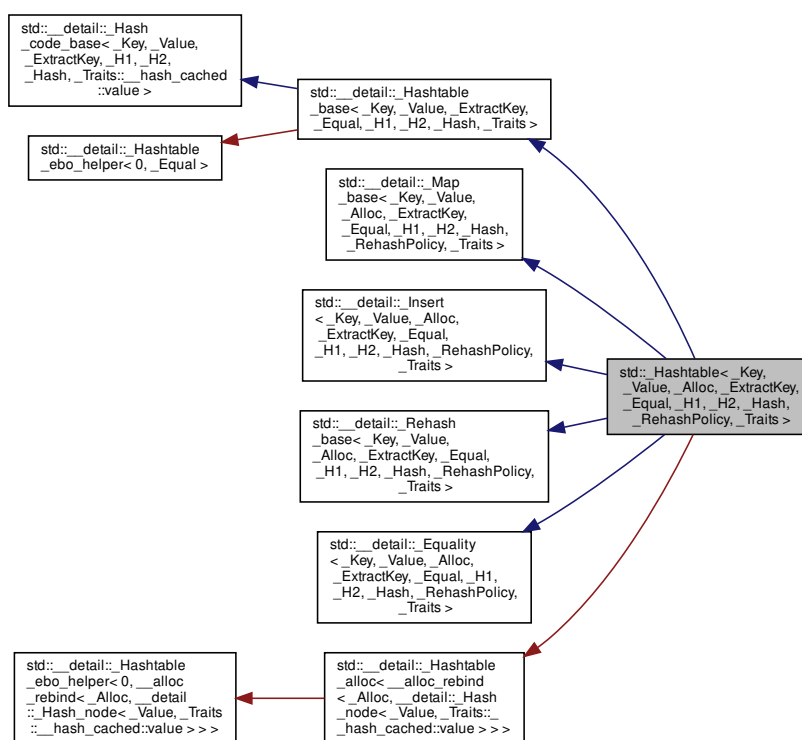
Definition at line 229 of file hashtable\_policy.h.

- hashtable policy.h

#### 4.116 std::\_Hashtable<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

##### Class Template Reference

Inheritance diagram for `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, Traits >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- using **const\_iterator** = typename `__hashtable_base::const_iterator`
- using **const\_local\_iterator** = typename `__hashtable_base::const_local_iterator`
- typedef `__value_alloc_traits::const_pointer` **const\_pointer**
- typedef const value\_type & **const\_reference**
- using **difference\_type** = typename `__hashtable_base::difference_type`
- using **iterator** = typename `__hashtable_base::iterator`
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = typename `__hashtable_base::local_iterator`
- typedef `__value_alloc_traits::pointer` **pointer**
- typedef value\_type & **reference**
- using **size\_type** = typename `__hashtable_base::size_type`
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Hashtable** (size\_type \_\_bkt\_count\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- template<typename \_InputIterator >  
**\_Hashtable** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_bkt\_count\_hint, const \_H1 &, const \_H2 &, const \_Hash &, const \_Equal &, const \_ExtractKey &, const allocator\_type &)
- **\_Hashtable** (const [\\_Hashtable](#) &)
- **\_Hashtable** ([\\_Hashtable](#) &&) noexcept
- **\_Hashtable** (const [\\_Hashtable](#) &, const allocator\_type &)
- **\_Hashtable** ([\\_Hashtable](#) &&, const allocator\_type &)
- **\_Hashtable** (const allocator\_type & \_\_a)
- **\_Hashtable** (size\_type \_\_bkt\_count\_hint, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- template<typename \_InputIterator >  
**\_Hashtable** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_bkt\_count\_hint=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- **\_Hashtable** (initializer\_list< value\_type > \_\_l, size\_type \_\_bkt\_count\_hint=0, const \_H1 & \_\_hf=\_H1(), const key\_equal & \_\_eq=key\_equal(), const allocator\_type & \_\_a=allocator\_type())
- const \_RehashPolicy & **\_\_rehash\_policy** () const
- void **\_\_rehash\_policy** (const \_RehashPolicy & \_\_pol)
- template<typename... \_Args>  
auto **\_M\_emplace** (true\_type, \_Args &&... \_\_args) -> [pair](#)< iterator, bool >
- template<typename... \_Args>  
auto **\_M\_emplace** (const\_iterator \_\_hint, false\_type, \_Args &&... \_\_args) -> iterator
- template<typename \_Arg, typename \_NodeGenerator >  
auto **\_M\_insert** (\_Arg && \_\_v, const \_NodeGenerator & \_\_node\_gen, true\_type, size\_type \_\_n\_elt) -> [pair](#)< iterator, bool >
- template<typename \_Arg, typename \_NodeGenerator >  
auto **\_M\_insert** (const\_iterator \_\_hint, \_Arg && \_\_v, const \_NodeGenerator & \_\_node\_gen, false\_type) -> iterator
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- local\_iterator **begin** (size\_type \_\_bkt)
- const\_local\_iterator **begin** (size\_type \_\_bkt) const

- `size_type bucket` (const key\_type &\_\_k) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (size\_type \_\_bkt) const
- `const_iterator cbegin` () const noexcept
- `const_local_iterator cbegin` (size\_type \_\_bkt) const
- `const_iterator cend` () const noexcept
- `const_local_iterator cend` (size\_type \_\_bkt) const
- `void clear` () noexcept
- `size_type count` (const key\_type &\_\_k) const
- `template<typename... _Args>`  
`__ireturn_type emplace` (\_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator emplace_hint` (const\_iterator \_\_hint, \_Args &&... \_\_args)
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `local_iterator end` (size\_type \_\_bkt)
- `const_local_iterator end` (size\_type \_\_bkt) const
- `std::pair< iterator, iterator > equal_range` (const key\_type &\_\_k)
- `std::pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_k) const
- `iterator erase` (const\_iterator)
- `iterator erase` (iterator \_\_it)
- `size_type erase` (const key\_type &\_\_k)
- `iterator erase` (const\_iterator, const\_iterator)
- `iterator find` (const key\_type &\_\_k)
- `const_iterator find` (const key\_type &\_\_k) const
- `allocator_type get_allocator` () const noexcept
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `size_type max_size` () const noexcept
- `_Hashtable & operator=` (const \_Hashtable &\_\_ht)
- `_Hashtable & operator=` (\_Hashtable &&\_\_ht) noexcept(\_\_node\_alloc\_traits::\_S\_nothrow\_move() &&is\_nothrow\_move\_assignable<\_H1 >::value &&is\_nothrow\_move\_assignable<\_Equal >::value)
- `_Hashtable & operator=` (initializer\_list< value\_type > \_\_l)
- `void rehash` (size\_type \_\_bkt\_count)
- `size_type size` () const noexcept
- `void swap` (\_Hashtable &) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_H1 >, \_\_is\_nothrow\_swappable<\_Equal >>::value)

#### Protected Member Functions

- `size_type _M_bucket_index` (\_\_node\_type \*\_\_n) const noexcept
- `size_type _M_bucket_index` (const key\_type &\_\_k, \_\_hash\_code \_\_c) const
- `template<typename... _Args>`  
`std::pair< iterator, bool > _M_emplace` (true\_type, \_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator _M_emplace` (false\_type \_\_uk, \_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator _M_emplace` (const\_iterator, true\_type \_\_uk, \_Args &&... \_\_args)

- `template<typename... _Args>`  
`iterator _M_emplace (const_iterator, false\_type, _Args &&... __args)`
- `const _Equal & _M_eq () const`
- `bool _M_equals (const_Key &__k, __hash_code __c, \_\_node\_type *__n) const`
- `size_type _M_erase (true\_type, const key_type &)`
- `size_type _M_erase (false\_type, const key_type &)`
- `iterator _M_erase (size_type __bkt, __node_base *__prev_n, \_\_node\_type *__n)`
- `__node_base * _M_find_before_node (size_type, const key_type &, __hash_code) const`
- `\_\_node\_type * _M_find_node (size_type __bkt, const key_type &__key, __hash_code __c) const`
- `__node_base * _M_get_previous_node (size_type __bkt, __node_base *__n)`
- `template<typename _Arg, typename _NodeGenerator >`  
`std::pair< iterator, bool > _M_insert (_Arg &&, const _NodeGenerator &, true\_type, size_type=1)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (_Arg &&__arg, const _NodeGenerator &__node_gen, false\_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (const_iterator, _Arg &&__arg, const _NodeGenerator &__node_gen, true\_type __uk)`
- `template<typename _Arg, typename _NodeGenerator >`  
`iterator _M_insert (const_iterator, _Arg &&, const _NodeGenerator &, false\_type)`
- `void _M_insert_bucket_begin (size_type, \_\_node\_type *)`
- `iterator _M_insert_multi_node (\_\_node\_type *__hint, const key_type &__k, __hash_code __code, \_\_node\_type *__n)`
- `iterator _M_insert_unique_node (const key_type &__k, size_type __bkt, __hash_code __code, \_\_node\_type *__n, size_type __n_elt=1)`
- `void _M_remove_bucket_begin (size_type __bkt, \_\_node\_type *__next_n, size_type __next_bkt)`
- `void _M_swap (_Hashtable_base &__x)`

## Private Types

- `using __bucket_alloc_traits = std::allocator\_traits< __bucket_alloc_type >`
- `using __bucket_alloc_type = __alloc_rebind< __node_alloc_type, __bucket_type >`

## Private Member Functions

- `__bucket_type * _M_allocate_buckets (std::size_t __bkt_count)`
- `\_\_node\_type * _M_allocate_node (_Args &&... __args)`
- `auto _M_allocate_node (_Args &&... __args) -> \_\_node\_type *`
- `void _M_deallocate_buckets (__bucket_type *, std::size_t __bkt_count)`
- `void _M_deallocate_node (\_\_node\_type *__n)`
- `void _M_deallocate_node_ptr (\_\_node\_type *__n)`
- `void _M_deallocate_nodes (\_\_node\_type *__n)`
- `__node_alloc_type & _M_node_allocator ()`
- `const __node_alloc_type & _M_node_allocator () const`



## Friends

- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Unique_keysa>`  
`struct __detail:: Equality`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Constant_iteratorsa>`  
`struct __detail:: Insert`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa >`  
`struct __detail:: Insert_base`
- `template<typename _Keya, typename _Valuea, typename _Alloca, typename _ExtractKeya, typename _Equala, typename _H1a, typename _H2a, typename _Hasha, typename _RehashPolicya, typename _Traitsa, bool _Unique_keysa>`  
`struct __detail:: Map_base`

### 4.116.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
class std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Hashtable`.

#### Template Parameters

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Value</code>        | CopyConstructible type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>_Key</code>          | CopyConstructible type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>_Alloc</code>        | An allocator type ([lib.allocator.requirements]) whose <code>_Alloc::value_type</code> is <code>_Value</code> . As a conforming extension, we allow for <code>_Alloc::value_type != _Value</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>_ExtractKey</code>   | Function object that takes an object of type <code>_Value</code> and returns a value of type <code>_Key</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>_Equal</code>        | Function object that takes two objects of type <code>k</code> and returns a bool-like value that is true if the two objects are considered equal.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>_H1</code>           | The hash function. A unary function object with argument type <code>_Key</code> and result type <code>size_t</code> . Return values should be distributed over the entire range <code>[0, numeric_limits&lt;size_t&gt;::max())</code> .                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>_H2</code>           | The range-hashing function (in the terminology of Tavori and Dreizin). A binary function object whose argument types and result type are all <code>size_t</code> . Given arguments <code>r</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> .                                                                                                                                                                                                                                                                                                                                                                                |
| <code>_Hash</code>         | The ranged hash function (Tavori and Dreizin). A binary function whose argument types are <code>_Key</code> and <code>size_t</code> and whose result type is <code>size_t</code> . Given arguments <code>k</code> and <code>N</code> , the return value is in the range <code>[0, N)</code> . Default: <code>hash(k, N) = h2(h1(k), N)</code> . If <code>_Hash</code> is anything other than the default, <code>_H1</code> and <code>_H2</code> are ignored.                                                                                                                                                                                                 |
| <code>_RehashPolicy</code> | Policy class with three members, all of which govern the bucket count. <code>_M_next_bkt(n)</code> returns a bucket count no smaller than <code>n</code> . <code>_M_bkt_for_elements(n)</code> returns a bucket count appropriate for an element count of <code>n</code> . <code>_M_need_rehash(n_bkt, n_elt, n_ins)</code> determines whether, if the current bucket count is <code>n_bkt</code> and the current element count is <code>n_elt</code> , we need to increase the bucket count. If so, returns <code>make_pair(true, n)</code> , where <code>n</code> is the new bucket count. If not, returns <code>make_pair(false, &lt;anything&gt;)</code> |
| <code>_Traits</code>       | Compile-time class with three boolean <code>std::integral_constant</code> members: <code>__cache_hash_code</code> , <code>__constant_iterators</code> , <code>__unique_keys</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Each `_Hashtable` data structure has:

- `_Bucket[] _M_buckets`
- `_Hash_node_base _M_before_begin`
- `size_type _M_bucket_count`
- `size_type _M_element_count`

with `_Bucket` being `_Hash_node*` and `_Hash_node` containing:

- `_Hash_node* _M_next`
- `Tp _M_value`
- `size_t _M_hash_code` if `cache_hash_code` is true

In terms of Standard containers the hashtable is like the aggregation of:

- `std::forward_list<_Node>` containing the elements
- `std::vector<std::forward_list<_Node>::iterator>` representing the buckets

The non-empty buckets contain the node before the first node in the bucket. This design makes it possible to implement something like a `std::forward_list::insert_after` on container insertion and `std::forward_list::erase_after` on container erase calls. `_M_before_begin` is equivalent to `std::forward_list::before_begin`. Empty buckets contain `nullptr`. Note that one of the non-empty buckets contains `&_M_before_begin` which is not a dereferenceable node so the node pointer in a bucket shall never be dereferenced, only its next node can be.

Walking through a bucket's nodes requires a check on the hash code to see if each node is still in the bucket. Such a design assumes a quite efficient hash functor and is one of the reasons it is highly advisable to set `__cache_hash_code` to true.

The container iterators are simply built from nodes. This way incrementing the iterator is perfectly efficient independent of how many empty buckets there are in the container.

On insert we compute the element's hash code and use it to find the bucket index. If the element must be inserted in an empty bucket we add it at the beginning of the singly linked list and make the bucket point to `_M_before_begin`. The bucket that used to point to `_M_before_begin`, if any, is updated to point to its new before begin node.

On erase, the simple iterator design requires using the hash functor to get the index of the bucket to update. For this reason, when `__cache_hash_code` is set to false the hash functor must not throw and this is enforced by a static assertion.

Functionality is implemented by decomposition into base classes, where the derived `_Hashtable` class is used in `↵` `Map_base`, `_Insert`, `_Rehash_base`, and `_Equality` base classes to access the "this" pointer. `_Hashtable_base` is used in the base classes as a non-recursive, fully-completed-type so that detailed nested type information, such as iterator type and node type, can be used. This is similar to the "Curiously Recurring Template Pattern" (CRTP) technique, but uses a reconstructed, not explicitly passed, template pattern.

Base class templates are:

- `__detail::_Hashtable_base`
- `__detail::_Map_base`
- `__detail::_Insert`
- `__detail::_Rehash_base`
- `__detail::_Equality`

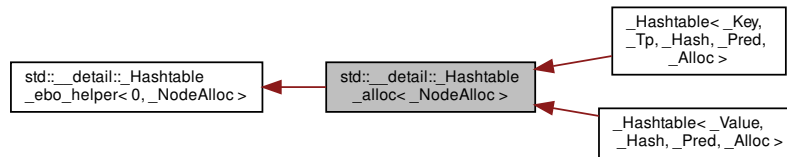
Definition at line 173 of file `bits/hashtable.h`.

The documentation for this class was generated from the following file:

- [bits/hashtable.h](#)

#### 4.117 std::\_\_detail::\_Hashtable\_alloc< \_NodeAlloc > Struct Template Reference

Inheritance diagram for `std::__detail::_Hashtable_alloc< _NodeAlloc >`:



#### Public Types

- using `__bucket_alloc_traits` = `std::allocator_traits< __bucket_alloc_type >`
- using `__bucket_alloc_type` = `__alloc_rebind< __node_alloc_type, __bucket_type >`
- using `__bucket_type` = `__node_base *`
- using `__node_alloc_traits` = `__gnu_cxx::__alloc_traits< __node_alloc_type >`
- using `__node_alloc_type` = `_NodeAlloc`
- using `__node_base` = `__detail::_Hash_node_base`
- using `__node_type` = `typename _NodeAlloc::value_type`
- using `__value_alloc_traits` = `typename __node_alloc_traits::template rebind_traits< typename __node_type::value_type >`

## Public Member Functions

- `_Hashtable_alloc` (const `_Hashtable_alloc` &)=default
- `_Hashtable_alloc` (`_Hashtable_alloc` &&)=default
- `template<typename _Alloc >`  
`_Hashtable_alloc` (`_Alloc` &&`_a`)
- `__bucket_type * _M_allocate_buckets` (std::size\_t `__bkt_count`)
- `template<typename... _Args>`  
`__node_type * _M_allocate_node` (`_Args` &&... `__args`)
- `template<typename... _Args>`  
`auto _M_allocate_node` (`_Args` &&... `__args`) -> `__node_type *`
- `void _M_deallocate_buckets` (`__bucket_type *`, std::size\_t `__bkt_count`)
- `void _M_deallocate_node` (`__node_type *` `__n`)
- `void _M_deallocate_node_ptr` (`__node_type *` `__n`)
- `void _M_deallocate_nodes` (`__node_type *` `__n`)
- `__node_alloc_type & _M_node_allocator` ()
- `const __node_alloc_type & _M_node_allocator` () const

## 4.117.1 Detailed Description

```
template<typename _NodeAlloc>
struct std::__detail::_Hashtable_alloc< _NodeAlloc >
```

This type deals with all allocation and keeps an allocator instance through inheritance to benefit from EBO when possible.

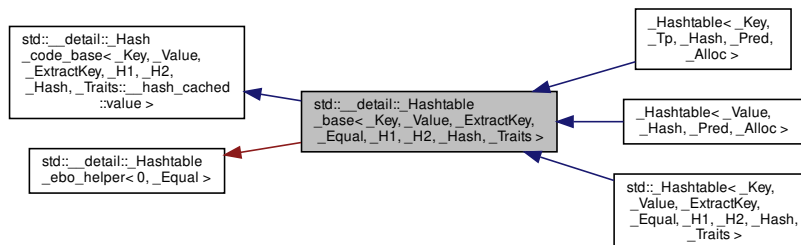
Definition at line 98 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.118 `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >` Struct Template Reference

Inheritance diagram for `std::__detail::_Hashtable_base< _Key, _Value, _ExtractKey, _Equal, _H1, _H2, _Hash, _Traits >`:



## Public Types

- using **\_\_constant\_iterators** = typename \_\_traits\_type::\_\_constant\_iterators
- using **\_\_hash\_cached** = typename \_\_traits\_type::\_\_hash\_cached
- using **\_\_hash\_code** = typename \_\_hash\_code\_base::\_\_hash\_code
- using **\_\_hash\_code\_base** = [\\_Hash\\_code\\_base](#)<\_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_hash\_↵cached::value>
- using **\_\_ireturn\_type** = typename `std::conditional`< \_\_unique\_keys::value, `std::pair`< iterator, bool >, iterator>::type
- using **\_\_node\_type** = typename \_\_hash\_code\_base::\_\_node\_type
- using **\_\_traits\_type** = \_Traits
- using **\_\_unique\_keys** = typename \_\_traits\_type::\_\_unique\_keys
- using **const\_iterator** = [\\_\\_detail::\\_\\_Node\\_const\\_iterator](#)< value\_type, \_\_constant\_iterators::value, \_\_hash\_↵cached::value>
- using **const\_local\_iterator** = [\\_\\_detail::\\_\\_Local\\_const\\_iterator](#)< key\_type, value\_type, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators::value, \_\_hash\_cached::value>
- typedef `std::ptrdiff_t` **difference\_type**
- using **iterator** = [\\_\\_detail::\\_\\_Node\\_iterator](#)< value\_type, \_\_constant\_iterators::value, \_\_hash\_cached::value>
- typedef `_Equal` **key\_equal**
- typedef `_Key` **key\_type**
- using **local\_iterator** = [\\_\\_detail::\\_\\_Local\\_iterator](#)< key\_type, value\_type, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_↵constant\_iterators::value, \_\_hash\_cached::value>
- typedef `std::size_t` **size\_type**
- typedef `_Value` **value\_type**

## Protected Member Functions

- **\_Hashtable\_base** (const \_ExtractKey &\_\_ex, const \_H1 &\_\_h1, const \_H2 &\_\_h2, const \_Hash &\_\_hash, const \_Equal &\_\_eq)
- const \_Equal & **\_M\_eq** () const
- bool **\_M\_equals** (const \_Key &\_\_k, \_\_hash\_code \_\_c, \_\_node\_type \*\_\_n) const
- void **\_M\_swap** ([\\_Hashtable\\_base](#) &\_\_x)

### 4.118.1 Detailed Description

template<typename \_Key, typename \_Value, typename \_ExtractKey, typename \_Equal, typename \_H1, typename \_H2, typename \_Hash, typename \_Traits>

struct `std::__detail::__Hashtable_base`<\_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits>

Primary class template `_Hashtable_base`.

Helper class adding management of `_Equal` functor to `_Hash_code_base` type.

Base class templates are:

- `__detail::__Hash_code_base`
- `__detail::__Hashtable_ebo_helper`

Definition at line 58 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.119 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, __use_ebo >` Struct Template Reference

##### 4.119.1 Detailed Description

```
template<int _Nm, typename _Tp, bool __use_ebo = !_is_final(_Tp) && __is_empty(_Tp)>
struct std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, __use_ebo >
```

Primary class template `_Hashtable_ebo_helper`.

Helper class using EBO when it is not forbidden (the type is not final) and when it is worth it (the type is empty.)

Definition at line 1105 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.120 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false >` Struct Template Reference

##### Public Member Functions

- `template<typename _OtherTp >`  
`_Hashtable_ebo_helper (_OtherTp &&__tp)`
- `const _Tp & _M_cget () const`
- `_Tp & _M_get ()`

##### 4.120.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, false >
```

Specialization not using EBO.

Definition at line 1125 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.121 `std::__detail::_Hashtable_ebo_helper<_Nm, _Tp, true >` Struct Template Reference

Inherits `_Tp`.

## Public Member Functions

- `template<typename _OtherTp >  
_Hashtable_ebo_helper (_OtherTp &&__tp)`
- `const _Tp & _M_cget () const`
- `_Tp & _M_get ()`

### 4.121.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::__detail::__Hashtable_ebo_helper< _Nm, _Tp, true >
```

Specialization using EBO.

Definition at line 1109 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.122 std::\_\_detail::\_\_Hashtable\_traits< \_Cache\_hash\_code, \_Constant\_iterators, \_Unique\_keys > Struct Template Reference

### Public Types

- using `__constant_iterators` = `__bool_constant< _Constant_iterators >`
- using `__hash_cached` = `__bool_constant< _Cache_hash_code >`
- using `__unique_keys` = `__bool_constant< _Unique_keys >`

### 4.122.1 Detailed Description

```
template<bool _Cache_hash_code, bool _Constant_iterators, bool _Unique_keys>
struct std::__detail::__Hashtable_traits< _Cache_hash_code, _Constant_iterators, _Unique_keys >
```

struct \_Hashtable\_traits

Important traits for hash tables.

### Template Parameters

|                                  |                                                                                                                                                                                                                                                                                                          |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_Cache_hash_code</code>    | Boolean value. True if the value of the hash function is stored along with the value. This is a time-space tradeoff. Storing it may improve lookup speed by reducing the number of times we need to call the <code>_Hash</code> or <code>_Equal</code> functors.                                         |
| <code>_Constant_iterators</code> | Boolean value. True if iterator and <code>const_iterator</code> are both constant iterator types. This is true for <code>unordered_set</code> and <code>unordered_multiset</code> , false for <code>unordered_map</code> and <code>unordered_multimap</code> .                                           |
| <code>_Unique_keys</code>        | Boolean value. True if the return value of <code>_Hashtable::count(k)</code> is always at most one, false if it may be an arbitrary number. This is true for <code>unordered_set</code> and <code>unordered_map</code> , false for <code>unordered_multiset</code> and <code>unordered_multimap</code> . |
| Generated by Doxygen             |                                                                                                                                                                                                                                                                                                          |

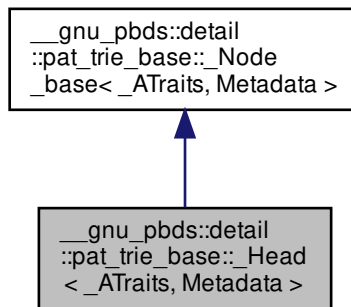
Definition at line 199 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.123 `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata >`:



##### Public Types

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits >::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `_Node_base<_ATraits, Metadata >` **base\_type**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `base_type::type_traits` **type\_traits**

##### Public Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_min**
- node\_pointer **m\_p\_parent**
- const `node_type` **m\_type**



## 4.123.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Head<_ATraits, Metadata>
```

Head node for PATRICIA tree.

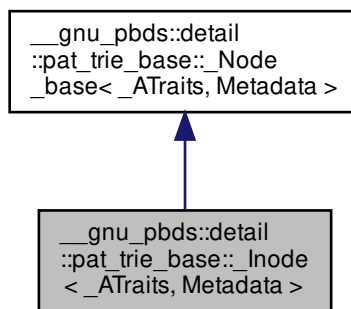
Definition at line 131 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.124 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>`:



## Classes

- struct [const\\_iterator](#)
- struct [iterator](#)

## Public Types

- enum { **arr\_size** }
- typedef [detail::rebind\\_traits](#)<\_Alloc, node\_pointer> **\_\_rebind\_np**
- typedef base\_type::allocator\_type **\_Alloc**
- typedef base\_type::access\_traits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef [\\_Node\\_base](#)<\_ATraits, Metadata> **base\_type**
- typedef \_\_rebind\_np::pointer **node\_pointer\_pointer**
- typedef \_\_rebind\_np::reference **node\_pointer\_reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef base\_type::type\_traits **type\_traits**
- typedef type\_traits::value\_type **value\_type**

## Public Member Functions

- **\_Inode** (size\_type, const a\_const\_iterator)
- node\_pointer **add\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- **const\_iterator begin** () const
- **iterator begin** ()
- **const\_iterator end** () const
- **iterator end** ()
- **iterator get\_child\_it** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- node\_const\_pointer **get\_child\_node** (a\_const\_iterator, a\_const\_iterator, a\_const\_pointer) const
- size\_type **get\_e\_ind** () const
- node\_const\_pointer **get\_join\_child** (node\_const\_pointer, a\_const\_pointer) const
- node\_pointer **get\_join\_child** (node\_pointer, a\_const\_pointer)
- node\_pointer **get\_lower\_bound\_child\_node** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer)
- leaf\_pointer **leftmost\_descendant** ()
- leaf\_const\_pointer **leftmost\_descendant** () const
- a\_const\_iterator **pref\_b\_it** () const
- a\_const\_iterator **pref\_e\_it** () const
- void **remove\_child** (node\_pointer)
- void **remove\_child** (iterator)
- void **replace\_child** (node\_pointer, a\_const\_iterator, a\_const\_iterator, a\_const\_pointer)
- leaf\_pointer **rightmost\_descendant** ()
- leaf\_const\_pointer **rightmost\_descendant** () const
- bool **should\_be\_mine** (a\_const\_iterator, a\_const\_iterator, size\_type, a\_const\_pointer) const
- void **update\_prefixes** (a\_const\_pointer)

## Public Attributes

- node\_pointer **m\_p\_parent**
- const **node\_type m\_type**

### 4.124.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >
```

Internal node type, PATRICIA tree.

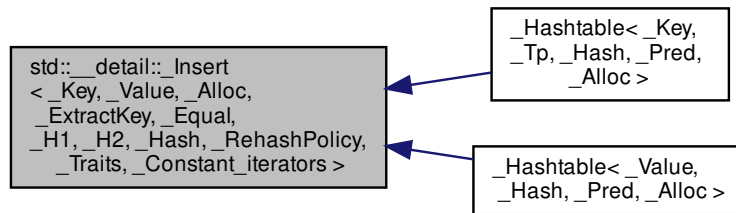
Definition at line 211 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.125 `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>` Struct Template Reference

Inheritance diagram for `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>`:



##### 4.125.1 Detailed Description

template<typename `_Key`, typename `_Value`, typename `_Alloc`, typename `_ExtractKey`, typename `_Equal`, typename `_H1`, typename `_H2`, typename `_Hash`, typename `_RehashPolicy`, typename `_Traits`, bool `_Constant_iterators` = `_Traits::__constant_iterators::value`>  
struct `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Constant_iterators>`

Primary class template `_Insert`.

Defines `insert` member functions that depend on `_Hashtable` policies, via partial specializations.

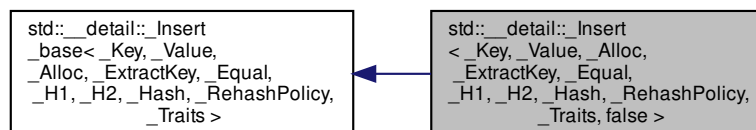
Definition at line 935 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.126 `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false>` Struct Template Reference

Inheritance diagram for `std::__detail::_Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false>`:



## Public Types

- using **\_\_base\_type** = [\\_Insert\\_base](#)< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Rehash, \_Policy, \_Traits >
- using **\_\_hashtable** = typename [\\_\\_base\\_type::\\_\\_hashtable](#)
- using **\_\_ireturn\_type** = typename [\\_\\_base\\_type::\\_\\_ireturn\\_type](#)
- template<typename \_Pair >  
using **\_\_is\_cons** = [std::is\\_constructible](#)< value\_type, \_Pair && >
- using **\_\_unique\_keys** = typename [\\_\\_base\\_type::\\_\\_unique\\_keys](#)
- template<typename \_Pair >  
using **\_\_IFcons** = [std::enable\\_if](#)< [\\_\\_is\\_cons](#)< \_Pair >::value >
- template<typename \_Pair >  
using **\_\_IFcons\_p** = typename [\\_\\_IFcons](#)< \_Pair >::type
- using **const\_iterator** = typename [\\_\\_base\\_type::const\\_iterator](#)
- using **iterator** = typename [\\_\\_base\\_type::iterator](#)
- using **value\_type** = typename [\\_\\_base\\_type::value\\_type](#)

## Public Member Functions

- [\\_\\_ireturn\\_type](#) **insert** (const value\_type &\_\_v)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_v)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **insert** ([initializer\\_list](#)< value\_type > \_\_l)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_v)
- [\\_\\_ireturn\\_type](#) **insert** (const value\_type &\_\_v)
- template<typename \_Pair, typename = [\\_\\_IFcons\\_p](#)< \_Pair >>  
[\\_\\_ireturn\\_type](#) **insert** (\_Pair && \_\_v)
- template<typename \_Pair, typename = [\\_\\_IFcons\\_p](#)< \_Pair >>  
iterator **insert** (const\_iterator \_\_hint, \_Pair && \_\_v)

## Protected Types

- using **\_\_hashtable\_base** = [\\_Hashtable\\_base](#)< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using **\_\_node\_alloc\_type** = [\\_alloc\\_rebind](#)< \_Alloc, [\\_\\_node\\_type](#) >
- using **\_\_node\_gen\_type** = [\\_AllocNode](#)< [\\_\\_node\\_alloc\\_type](#) >
- using **\_\_node\_type** = [\\_Hash\\_node](#)< \_Value, \_Traits::\_\_hash\_cached::value >
- using **size\_type** = typename [\\_\\_hashtable\\_base::size\\_type](#)

## Protected Member Functions

- [\\_\\_hashtable](#) & **\_M\_conjure\_hashtable** ()
- template<typename \_InputIterator, typename \_NodeGetter >  
void **\_M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [true\\_type](#))
- template<typename \_InputIterator, typename \_NodeGetter >  
void **\_M\_insert\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_NodeGetter &, [false\\_type](#))

#### 4.126.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Specialization.

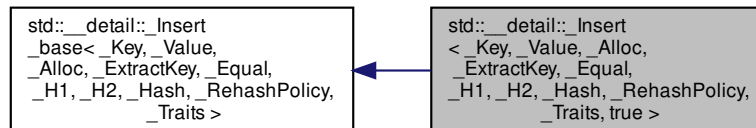
Definition at line 989 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.127 `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

Inheritance diagram for `std::__detail::Insert<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`:



#### Public Types

- using **\_\_base\_type** = [\\_Insert\\_base](#)<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- using **\_\_hashtable** = typename [\\_\\_base\\_type::\\_\\_hashtable](#)
- using **\_\_hashtable\_base** = [\\_Hashtable\\_base](#)<\_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- using **\_\_ireturn\_type** = typename [\\_\\_hashtable\\_base::\\_\\_ireturn\\_type](#)
- using **\_\_node\_gen\_type** = typename [\\_\\_base\\_type::\\_\\_node\\_gen\\_type](#)
- using **\_\_unique\_keys** = typename [\\_\\_base\\_type::\\_\\_unique\\_keys](#)
- using **const\_iterator** = typename [\\_\\_base\\_type::const\\_iterator](#)
- using **iterator** = typename [\\_\\_base\\_type::iterator](#)
- using **value\_type** = typename [\\_\\_base\\_type::value\\_type](#)

## Public Member Functions

- `__ireturn_type insert (const value_type &__v)`
- `iterator insert (const_iterator __hint, const value_type &__v)`
- `void insert (initializer_list< value_type > __l)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `iterator insert (const_iterator __hint, const value_type &__v)`
- `__ireturn_type insert (const value_type &__v)`
- `__ireturn_type insert (value_type &&__v)`
- `iterator insert (const_iterator __hint, value_type &&__v)`

## Protected Types

- `using __node_alloc_type = __alloc_rebind< _Alloc, __node_type >`
- `using __node_type = _Hash_node< _Value, _Traits::__hash_cached::value >`
- `using size_type = typename __hashtable_base::size_type`

## Protected Member Functions

- `__hashtable & _M_conjure_hashtable ()`
- `template<typename _InputIterator, typename _NodeGetter >`  
`void _M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, true_type)`
- `template<typename _InputIterator, typename _NodeGetter >`  
`void _M_insert_range (_InputIterator __first, _InputIterator __last, const _NodeGetter &, false_type)`

### 4.127.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::Insert< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Specialization.

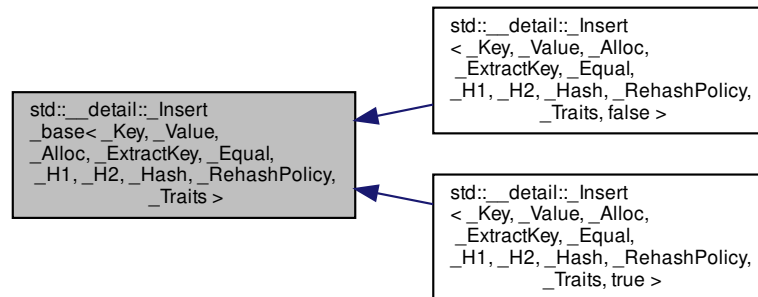
Definition at line 942 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.128 `std::__detail::__Insert_base<_Key,_Value,_Alloc,_ExtractKey,_Equal,_H1,_H2,_Hash,_RehashPolicy,_Traits>` Struct Template Reference

Inheritance diagram for `std::__detail::__Insert_base<_Key,_Value,_Alloc,_ExtractKey,_Equal,_H1,_H2,_Hash,_RehashPolicy,_Traits>`:



#### Public Member Functions

- `__ireturn_type insert` (const value\_type &\_\_v)
- iterator `insert` (const\_iterator \_\_hint, const value\_type &\_\_v)
- void `insert` (initializer\_list<value\_type> \_\_l)
- template<typename \_\_InputIterator>  
void `insert` (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last)

#### Protected Types

- using `__hashtable` = `__Hashtable`<\_Key,\_Value,\_Alloc,\_ExtractKey,\_Equal,\_H1,\_H2,\_Hash,\_RehashPolicy,\_Traits>
- using `__hashtable_base` = `__Hashtable_base`<\_Key,\_Value,\_ExtractKey,\_Equal,\_H1,\_H2,\_Hash,\_Traits>
- using `__ireturn_type` = typename `__hashtable_base::__ireturn_type`
- using `__node_alloc_type` = `__alloc_rebind`<\_Alloc, `__node_type`>
- using `__node_gen_type` = `__AllocNode`<\_\_node\_alloc\_type>
- using `__node_type` = `__Hash_node`<\_Value,\_Traits::\_\_hash\_cached::value>
- using `__unique_keys` = typename `__hashtable_base::__unique_keys`
- using `const_iterator` = typename `__hashtable_base::const_iterator`
- using `iterator` = typename `__hashtable_base::iterator`
- using `size_type` = typename `__hashtable_base::size_type`
- using `value_type` = typename `__hashtable_base::value_type`

#### Protected Member Functions

- `__hashtable` & `__M_conjure_hashtable` ()
- template<typename \_\_InputIterator, typename \_\_NodeGetter>  
void `__M_insert_range` (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last, const \_\_NodeGetter &, `true_type`)
- template<typename \_\_InputIterator, typename \_\_NodeGetter>  
void `__M_insert_range` (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last, const \_\_NodeGetter &, `false_type`)

#### 4.128.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Insert_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >
```

Primary class template `_Insert_base`.

Defines `insert` member functions appropriate to all `_Hashtables`.

Definition at line 798 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.129 `__gnu_cxx::_Invalid_type` Struct Reference

##### 4.129.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize `_Pointer_adapter` for cases of `void*`, `const void*`, and so on.

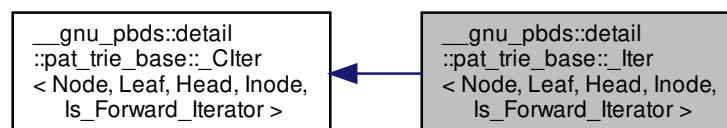
Definition at line 216 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

#### 4.130 `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >`:





## Public Types

- `typedef allocator_type _Alloc`
- `typedef base_type::allocator_type allocator_type`
- `typedef \_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > base_type`
- `typedef type_traits::const_pointer const_pointer`
- `typedef type_traits::const_reference const_reference`
- `typedef allocator_type::difference_type difference_type`
- `typedef base_type::head_pointer head_pointer`
- `typedef Inode::iterator inode_iterator`
- `typedef base_type::inode_pointer inode_pointer`
- `typedef std::bidirectional\_iterator\_tag iterator_category`
- `typedef base_type::leaf_const_pointer leaf_const_pointer`
- `typedef base_type::leaf_pointer leaf_pointer`
- `typedef base_type::node_pointer node_pointer`
- `typedef type_traits::pointer pointer`
- `typedef type_traits::reference reference`
- `typedef base_type::type_traits type_traits`
- `typedef type_traits::value_type value_type`

## Public Member Functions

- `\_Iter (node_pointer p_nd=0)`
- `\_Iter (const \_Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)`
- `bool operator!= (const \_CIter &other) const`
- `bool operator!= (const \_CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const`
- `reference operator* () const`
- `\_Iter & operator++ ()`
- `\_Iter operator++ (int)`
- `\_Iter & operator-- ()`
- `\_Iter operator-- (int)`
- `pointer operator-> () const`
- `\_Iter & operator= (const \_Iter &other)`
- `\_Iter & operator= (const \_Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)`
- `bool operator== (const \_CIter &other) const`
- `bool operator== (const \_CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const`

## Public Attributes

- `node_pointer m_p_nd`

## Protected Member Functions

- `void dec (false_type)`
- `void dec (true_type)`
- `void inc (false_type)`
- `void inc (true_type)`

### Static Protected Member Functions

- static node\_pointer **get\_larger\_sibling** (node\_pointer p\_nd)
- static node\_pointer **get\_smaller\_sibling** (node\_pointer p\_nd)
- static leaf\_pointer **leftmost\_descendant** (node\_pointer p\_nd)
- static leaf\_pointer **rightmost\_descendant** (node\_pointer p\_nd)

#### 4.130.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
```

Iterator.

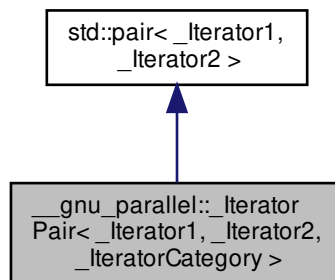
Definition at line 709 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

#### 4.131 \_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory > Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >:



### Public Types

- typedef [std::iterator\\_traits](#)< \_Iterator1 > **TraitsType**
- typedef TraitsType::difference\_type **difference\_type**
- typedef \_Iterator1 [first\\_type](#)
- typedef \_IteratorCategory **iterator\_category**
- typedef [\\_IteratorPair](#) \* **pointer**
- typedef [\\_IteratorPair](#) & **reference**
- typedef \_Iterator2 [second\\_type](#)
- typedef void **value\_type**

## Public Member Functions

- `_IteratorPair` (const `_Iterator1` &\_\_first, const `_Iterator2` &\_\_second)
- `operator _Iterator2` () const
- `_IteratorPair` `operator+` (difference\_type \_\_delta) const
- `_IteratorPair` & `operator++` ()
- const `_IteratorPair` `operator++` (int)
- difference\_type `operator-` (const `_IteratorPair` &\_\_other) const
- `_IteratorPair` & `operator--` ()
- const `_IteratorPair` `operator--` (int)
- `_IteratorPair` & `operator=` (const `_IteratorPair` &\_\_other)
- constexpr void `swap` (pair &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_Iterator1>, \_\_is\_nothrow\_swappable<\_Iterator2>>::value)

## Public Attributes

- `_Iterator1` `first`
- `_Iterator2` `second`

## Related Functions

(Note that these are not member functions.)

- constexpr `pair`< typename \_\_decay\_and\_strip<\_Iterator1>::\_\_type, typename \_\_decay\_and\_strip<\_Iterator2>::\_\_type > `make_pair` (\_Iterator1 &&\_\_x, \_Iterator2 &&\_\_y)
- constexpr enable\_if< \_\_and\_< \_\_is\_swappable<\_Iterator1>, \_\_is\_swappable<\_Iterator2>>::value >::type `swap` (pair<\_Iterator1, \_Iterator2> &\_\_x, pair<\_Iterator1, \_Iterator2> &\_\_y) noexcept(noexcept(\_\_x.swap(\_\_y)))
- constexpr bool `operator==` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)
- constexpr bool `operator<` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)
- constexpr bool `operator!=` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)
- constexpr bool `operator>` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)
- constexpr bool `operator<=` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)
- constexpr bool `operator>=` (const pair<\_Iterator1, \_Iterator2> &\_\_x, const pair<\_Iterator1, \_Iterator2> &\_\_y)

## 4.131.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair<_Iterator1, _Iterator2, _IteratorCategory>
```

A pair of iterators. The usual iterator operations are applied to both child iterators.

Definition at line 45 of file iterator.h.

#### 4.131.2 Member Typedef Documentation

##### 4.131.2.1 first\_type

```
typedef _Iterator1 std::pair< _Iterator1 , _Iterator2 >::first_type [inherited]
```

The type of the `first` member.

Definition at line 214 of file `stl_pair.h`.

##### 4.131.2.2 second\_type

```
typedef _Iterator2 std::pair< _Iterator1 , _Iterator2 >::second_type [inherited]
```

The type of the `second` member.

Definition at line 215 of file `stl_pair.h`.

#### 4.131.3 Member Function Documentation

##### 4.131.3.1 swap()

```
constexpr void std::pair< _Iterator1 , _Iterator2 >::swap (
 pair< _Iterator1, _Iterator2 > & __p) [inline], [noexcept], [inherited]
```

Swap the first members and then the second members.

Definition at line 439 of file `stl_pair.h`.

#### 4.131.4 Friends And Related Function Documentation

##### 4.131.4.1 make\_pair()

```
constexpr pair< typename __decay_and_strip< _Iterator1 >::__type, typename __decay_and_strip< ↵
 _Iterator2 >::__type > make_pair (
 _Iterator1 && __x,
 _Iterator2 && __y) [related]
```

A convenience wrapper for creating a pair from two objects.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | The first object.  |
| <code>__y</code> | The second object. |

#### Returns

A newly-constructed `pair<>` object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).

Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.

Definition at line 567 of file `stl_pair.h`.

#### 4.131.4.2 `operator!=(())`

```
constexpr bool operator!=(
 const pair<_Iterator1 , _Iterator2 > & __x,
 const pair<_Iterator1 , _Iterator2 > & __y) [related]
```

Uses `operator==` to find the result.

Definition at line 496 of file `stl_pair.h`.

#### 4.131.4.3 `operator<()`

```
constexpr bool operator<(
 const pair<_Iterator1 , _Iterator2 > & __x,
 const pair<_Iterator1 , _Iterator2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of the same type, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

Definition at line 489 of file `stl_pair.h`.

#### 4.131.4.4 operator<=()

```
constexpr bool operator<= (
 const pair< _Iterator1 , _Iterator2 > & __x,
 const pair< _Iterator1 , _Iterator2 > & __y) [related]
```

Uses operator< to find the result.

Definition at line 508 of file stl\_pair.h.

#### 4.131.4.5 operator==()

```
constexpr bool operator== (
 const pair< _Iterator1 , _Iterator2 > & __x,
 const pair< _Iterator1 , _Iterator2 > & __y) [related]
```

Two pairs of the same type are equal iff their members are equal.

Definition at line 466 of file stl\_pair.h.

#### 4.131.4.6 operator>()

```
constexpr bool operator> (
 const pair< _Iterator1 , _Iterator2 > & __x,
 const pair< _Iterator1 , _Iterator2 > & __y) [related]
```

Uses operator< to find the result.

Definition at line 502 of file stl\_pair.h.

#### 4.131.4.7 operator>=()

```
constexpr bool operator>= (
 const pair< _Iterator1 , _Iterator2 > & __x,
 const pair< _Iterator1 , _Iterator2 > & __y) [related]
```

Uses operator< to find the result.

Definition at line 514 of file stl\_pair.h.

#### 4.131.4.8 `swap()`

```
constexpr enable_if< __and< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > > &
::value >::type swap (
 pair< _Iterator1 , _Iterator2 > & __x,
 pair< _Iterator1 , _Iterator2 > & __y) [related]
```

Swap overload for pairs. Calls `std::pair::swap()`.

#### Note

This `std::swap` overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

Definition at line 533 of file `stl_pair.h`.

### 4.131.5 Member Data Documentation

#### 4.131.5.1 `first`

```
_Iterator1 std::pair< _Iterator1 , _Iterator2 >::first [inherited]
```

The first member.

Definition at line 217 of file `stl_pair.h`.

#### 4.131.5.2 `second`

```
_Iterator2 std::pair< _Iterator1 , _Iterator2 >::second [inherited]
```

The second member.

Definition at line 218 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [iterator.h](#)

### 4.132 `__gnu_parallel::_IteratorTriple<_Iterator1, _Iterator2, _Iterator3, _IteratorCategory>` Class Template Reference

#### Public Types

- typedef `std::iterator_traits<_Iterator1>::difference_type` **difference\_type**
- typedef `_IteratorCategory` **iterator\_category**
- typedef `_IteratorTriple` \* **pointer**
- typedef `_IteratorTriple` & **reference**
- typedef void **value\_type**

#### Public Member Functions

- **\_IteratorTriple** (const \_Iterator1 &\_\_first, const \_Iterator2 &\_\_second, const \_Iterator3 &\_\_third)
- **operator \_Iterator3** () const
- **\_IteratorTriple operator+** (difference\_type \_\_delta) const
- **\_IteratorTriple & operator++** ()
- const **\_IteratorTriple operator++** (int)
- difference\_type **operator-** (const **\_IteratorTriple** &\_\_other) const
- **\_IteratorTriple & operator--** ()
- const **\_IteratorTriple operator--** (int)
- **\_IteratorTriple & operator=** (const **\_IteratorTriple** &\_\_other)

#### Public Attributes

- \_Iterator1 **\_M\_first**
- \_Iterator2 **\_M\_second**
- \_Iterator3 **\_M\_third**

#### 4.132.1 Detailed Description

```
template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>
class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >
```

A triple of iterators. The usual iterator operations are applied to all three child iterators.

Definition at line 120 of file iterator.h.

The documentation for this class was generated from the following file:

- [iterator.h](#)

#### 4.133 \_\_gnu\_parallel::\_Job<\_DifferenceTp> Struct Template Reference

##### Public Types

- typedef \_DifferenceTp **\_DifferenceType**

##### Public Attributes

- volatile \_DifferenceType **\_M\_first**
- volatile \_DifferenceType **\_M\_last**
- volatile \_DifferenceType **\_M\_load**



#### 4.133.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::__Job<_DifferenceTp>
```

One \_\_job for a certain thread.

Definition at line 54 of file workstealing.h.

#### 4.133.2 Member Data Documentation

##### 4.133.2.1 \_\_M\_first

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::__M_first
```

First element.

Changed by owning and stealing thread. By stealing thread, always incremented.

Definition at line 62 of file workstealing.h.

Referenced by \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing().

##### 4.133.2.2 \_\_M\_last

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::__Job<_DifferenceTp>::__M_last
```

Last element.

Changed by owning thread only.

Definition at line 67 of file workstealing.h.

Referenced by \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing().

#### 4.133.2.3 `_M_load`

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load
```

Number of elements, i.e. `_M_last - _M_first + 1`.

Changed by owning thread only.

Definition at line 72 of file `workstealing.h`.

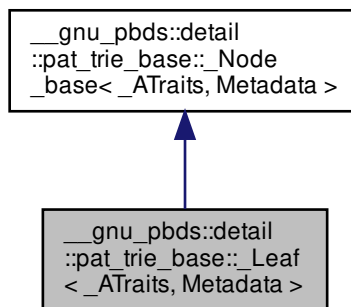
Referenced by `__gnu_parallel::_for_each_template_random_access_workstealing()`.

The documentation for this struct was generated from the following file:

- [workstealing.h](#)

#### 4.134 `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Leaf<_ATraits, Metadata>`:



#### Public Types

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits<_Alloc, _ATraits>::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `_Node_base<_ATraits, Metadata>` **base\_type**
- typedef `type_traits::const_reference` **const\_reference**
- typedef `detail::rebind_traits<_Alloc, _Node_base>::pointer` **node\_pointer**
- typedef `type_traits::reference` **reference**
- typedef `base_type::type_traits` **type\_traits**
- typedef `type_traits::value_type` **value\_type**

## Public Member Functions

- **\_Leaf** (const\_reference other)
- reference **value** ()
- const\_reference **value** () const

## Public Attributes

- node\_pointer **m\_p\_parent**
- const [node\\_type](#) **m\_type**

## 4.134.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf< ATraits, Metadata >
```

Leaf node for PATRICIA tree.

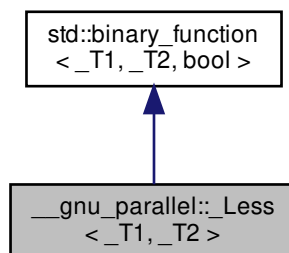
Definition at line 162 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.135 \_\_gnu\_parallel::\_Less&lt;\_T1,\_T2&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_Less<\_T1,\_T2>:



### Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _T1 &__t1`, `const _T2 &__t2`) `const`
- `bool` **operator()** (`const _T2 &__t2`, `const _T1 &__t1`) `const`

#### 4.135.1 Detailed Description

```
template<typename _T1, typename _T2>
struct __gnu_parallel::_Less< _T1, _T2 >
```

Similar to `std::less`, but allows two different types.

Definition at line 252 of file `base.h`.

#### 4.135.2 Member Typedef Documentation

##### 4.135.2.1 `first_argument_type`

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.135.2.2 `result_type`

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.135.2.3 `second_argument_type`

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

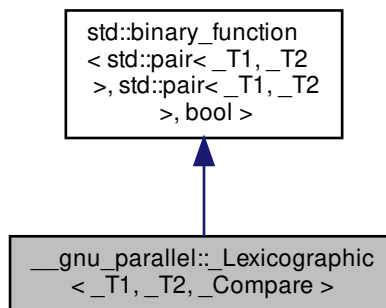
Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [base.h](#)

4.136 `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >`:



## Public Types

- typedef `std::pair< _T1, _T2 >` `first_argument_type`
- typedef `bool` `result_type`
- typedef `std::pair< _T1, _T2 >` `second_argument_type`

## Public Member Functions

- `_Lexicographic` (`_Compare &__comp`)
- `bool operator()` (`const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2`) `const`

#### 4.136.1 Detailed Description

```
template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >
```

Compare \_\_a pair of types lexicographically, ascending.

Definition at line 53 of file multiseq\_selection.h.

#### 4.136.2 Member Typedef Documentation

##### 4.136.2.1 first\_argument\_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2
> , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 4.136.2.2 result\_type

```
typedef bool std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2 > , bool >↔
::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

##### 4.136.2.3 second\_argument\_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 > , std::pair< _T1, _T2
> , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

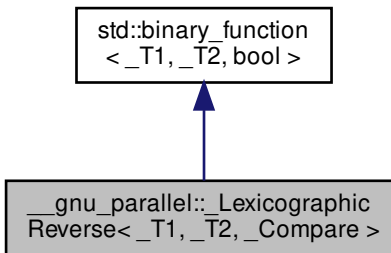
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

4.137 `__gnu_parallel::_LexicographicReverse<_T1,_T2,_Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LexicographicReverse<_T1,_T2,_Compare>`:



## Public Types

- typedef `_T1` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_T2` [second\\_argument\\_type](#)

## Public Member Functions

- `_LexicographicReverse` (`_Compare &__comp`)
- `bool operator()` (`const std::pair<_T1,_T2> &__p1, const std::pair<_T1,_T2> &__p2`) `const`

## 4.137.1 Detailed Description

```

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse<_T1,_T2,_Compare>

```

Compare `__a` pair of types lexicographically, descending.

Definition at line 80 of file `multiseq_selection.h`.

## 4.137.2 Member Typedef Documentation

#### 4.137.2.1 first\_argument\_type

```
typedef _T1 std::binary_function< _T1 , _T2 , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.137.2.2 result\_type

```
typedef bool std::binary_function< _T1 , _T2 , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.137.2.3 second\_argument\_type

```
typedef _T2 std::binary_function< _T1 , _T2 , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

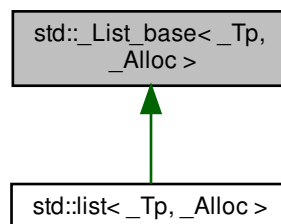
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [multiseq\\_selection.h](#)

### 4.138 std::\_List\_base< \_Tp, \_Alloc > Class Template Reference

Inheritance diagram for std::\_List\_base< \_Tp, \_Alloc >:





## Public Types

- typedef `_Alloc` **allocator\_type**

## Public Member Functions

- `_List_base` (const `_Node_alloc_type` &\_\_a) noexcept
- `_List_base` (`_List_base` &&)=default
- `_List_base` (`_List_base` &&\_\_x, `_Node_alloc_type` &&\_\_a)
- `_List_base` (`_Node_alloc_type` &&\_\_a, `_List_base` &&\_\_x)
- `_List_base` (`_Node_alloc_type` &&\_\_a)
- void `_M_clear` () noexcept
- `_Node_alloc_type` & `_M_get_Node_allocator` () noexcept
- const `_Node_alloc_type` & `_M_get_Node_allocator` () const noexcept
- void `_M_init` () noexcept
- void `_M_move_nodes` (`_List_base` &&\_\_x)

## Protected Types

- typedef `__gnu_cxx::__alloc_traits<_Node_alloc_type>` **\_Node\_alloc\_traits**
- typedef `_Tp_alloc_traits::template rebind<_List_node<_Tp>>::other` **\_Node\_alloc\_type**
- typedef `__gnu_cxx::__alloc_traits<_Tp_alloc_type>` **\_Tp\_alloc\_traits**
- typedef `__gnu_cxx::__alloc_traits<_Alloc>::template rebind<_Tp>::other` **\_Tp\_alloc\_type**

## Protected Member Functions

- void `_M_dec_size` (size\_t)
- size\_t `_M_distance` (const void \*, const void \*) const
- `_Node_alloc_traits::pointer` `_M_get_node` ()
- size\_t `_M_get_size` () const
- void `_M_inc_size` (size\_t)
- size\_t `_M_node_count` () const
- void `_M_put_node` (typename `_Node_alloc_traits::pointer` \_\_p) noexcept
- void `_M_set_size` (size\_t)

## Static Protected Member Functions

- static size\_t `_S_distance` (const `__detail::_List_node_base` \*\_\_first, const `__detail::_List_node_base` \*\_\_last)

## Protected Attributes

- `_List_impl` **\_M\_impl**

#### 4.138.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >
```

See bits/stl\_deque.h's `_Deque_base` for an explanation.

Definition at line 349 of file stl\_list.h.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

#### 4.139 `std::_List_const_iterator< _Tp >` Struct Template Reference

##### Public Types

- typedef const `_List_node`< \_Tp > `_Node`
- typedef `_List_const_iterator`< \_Tp > `_Self`
- typedef ptrdiff\_t `difference_type`
- typedef `_List_iterator`< \_Tp > `iterator`
- typedef `std::bidirectional_iterator_tag` `iterator_category`
- typedef const \_Tp \* `pointer`
- typedef const \_Tp & `reference`
- typedef \_Tp `value_type`

##### Public Member Functions

- `_List_const_iterator` (const `__detail::_List_node_base` \*\_\_x) noexcept
- `_List_const_iterator` (const `iterator` &\_\_x) noexcept
- `iterator_M_const_cast` () const noexcept
- reference `operator*` () const noexcept
- `_Self` & `operator++` () noexcept
- `_Self` `operator++` (int) noexcept
- `_Self` & `operator--` () noexcept
- `_Self` `operator--` (int) noexcept
- pointer `operator->` () const noexcept

##### Public Attributes

- const `__detail::_List_node_base` \* `_M_node`

##### Friends

- bool `operator!=` (const `_Self` &\_\_x, const `_Self` &\_\_y) noexcept
- bool `operator==` (const `_Self` &\_\_x, const `_Self` &\_\_y) noexcept

## 4.139.1 Detailed Description

```
template<typename _Tp>
struct std::_List_const_iterator<_Tp>
```

A `list::const_iterator`.

All the functions are op overloads.

Definition at line 74 of file `stl_iterator_base_funcs.h`.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

4.140 `std::_List_iterator<_Tp>` Struct Template Reference

## Public Types

- typedef `_List_node<_Tp>` **\_Node**
- typedef `_List_iterator<_Tp>` **\_Self**
- typedef `ptrdiff_t` **difference\_type**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**
- typedef `_Tp *` **pointer**
- typedef `_Tp &` **reference**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `_List_iterator` (`__detail::_List_node_base *`\_\_x) noexcept
- `_Self _M_const_cast` () const noexcept
- reference **operator\*** () const noexcept
- `_Self & operator++` () noexcept
- `_Self operator++` (int) noexcept
- `_Self & operator--` () noexcept
- `_Self operator--` (int) noexcept
- pointer **operator->** () const noexcept

## Public Attributes

- `__detail::_List_node_base *` **\_M\_node**

## Friends

- bool **operator!=** (const `_Self` &\_\_x, const `_Self` &\_\_y) noexcept
- bool **operator==** (const `_Self` &\_\_x, const `_Self` &\_\_y) noexcept

#### 4.140.1 Detailed Description

```
template<typename _Tp>
struct std::_List_iterator< _Tp >
```

A list::iterator.

All the functions are op overloads.

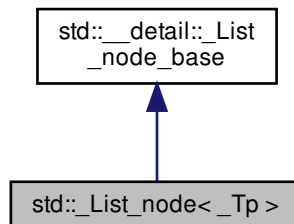
Definition at line 73 of file stl\_iterator\_base\_funcs.h.

The documentation for this struct was generated from the following files:

- [stl\\_iterator\\_base\\_funcs.h](#)
- [stl\\_list.h](#)

#### 4.141 std::\_List\_node< \_Tp > Struct Template Reference

Inheritance diagram for std::\_List\_node< \_Tp >:



##### Public Member Functions

- void **\_M\_hook** (`_List_node_base *const __position`) noexcept
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** (`_List_node_base *const __first`, `_List_node_base *const __last`) noexcept
- void **\_M\_unhook** () noexcept
- `_Tp * _M_valptr` ()
- `_Tp const * _M_valptr` () const

##### Static Public Member Functions

- static void **swap** (`_List_node_base & __x`, `_List_node_base & __y`) noexcept

## Public Attributes

- `_List_node_base * _M_next`
- `_List_node_base * _M_prev`
- `__gnu_cxx::__aligned_membuf<_Tp> _M_storage`

## 4.141.1 Detailed Description

```
template<typename _Tp>
struct std::_List_node<_Tp>
```

An actual node in the list.

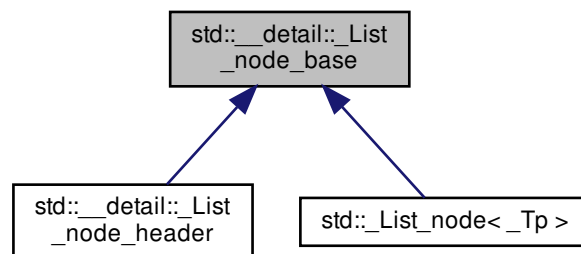
Definition at line 166 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

## 4.142 std::\_\_detail::\_List\_node\_base Struct Reference

Inheritance diagram for `std::__detail::_List_node_base`:



## Public Member Functions

- `void _M_hook (\_List\_node\_base *const __position) noexcept`
- `void _M_reverse () noexcept`
- `void _M_transfer (\_List\_node\_base *const __first, \_List\_node\_base *const __last) noexcept`
- `void _M_unhook () noexcept`

#### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

#### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 4.142.1 Detailed Description

Common part of a node in the list.

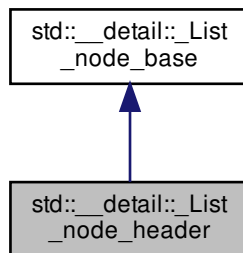
Definition at line 80 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.143 `std::__detail::_List_node_header` Struct Reference

Inheritance diagram for `std::__detail::_List_node_header`:



#### Public Member Functions

- **\_List\_node\_header** ([\\_List\\_node\\_header](#) &&\_\_x) noexcept
- void **\_M\_hook** ([\\_List\\_node\\_base](#) \*const \_\_position) noexcept
- void **\_M\_init** () noexcept
- void **\_M\_move\_nodes** ([\\_List\\_node\\_header](#) &&\_\_x)
- void **\_M\_reverse** () noexcept
- void **\_M\_transfer** ([\\_List\\_node\\_base](#) \*const \_\_first, [\\_List\\_node\\_base](#) \*const \_\_last) noexcept
- void **\_M\_unhook** () noexcept

#### Static Public Member Functions

- static void **swap** ([\\_List\\_node\\_base](#) &\_\_x, [\\_List\\_node\\_base](#) &\_\_y) noexcept

#### Public Attributes

- [\\_List\\_node\\_base](#) \* **\_M\_next**
- [\\_List\\_node\\_base](#) \* **\_M\_prev**

#### 4.143.1 Detailed Description

The list node header.

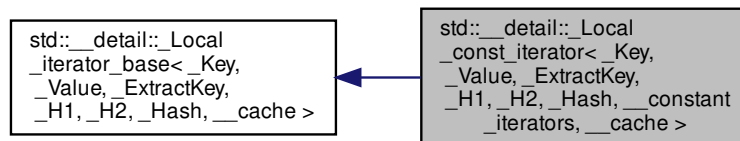
Definition at line 103 of file `stl_list.h`.

The documentation for this struct was generated from the following file:

- [stl\\_list.h](#)

#### 4.144 `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >` Struct Template Reference

Inheritance diagram for `std::__detail::__Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`:



#### Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `const _Value *` **pointer**
- typedef `const _Value &` **reference**
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Local\_const\_iterator** (const \_\_hash\_code\_base &\_\_base, [\\_Hash\\_node](#)< \_Value, \_\_cache > \*\_\_n, std::size\_t \_\_bkt, std::size\_t \_\_bkt\_count)
- **\_Local\_const\_iterator** (const [\\_Local\\_iterator](#)< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_ iterators, \_\_cache > &\_\_x)
- reference **operator\*** () const
- [\\_Local\\_const\\_iterator](#) & **operator++** ()
- [\\_Local\\_const\\_iterator](#) **operator++** (int)
- pointer **operator->** () const

## 4.144.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __constant_
iterators, bool __cache>
struct std::__detail::_Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_ iterators, __cache >
```

local const\_ iterators

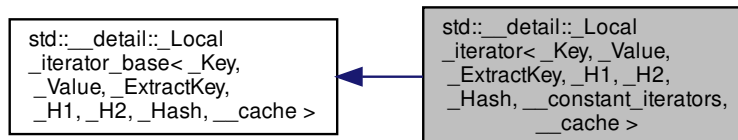
Definition at line 1657 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.145 std::\_\_detail::\_Local\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_ iterators, \_\_cache > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Local\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_ iterators, \_\_cache >:



## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef [std::conditional](#)< \_\_constant\_ iterators, const \_Value \*, \_Value \* >::type **pointer**
- typedef [std::conditional](#)< \_\_constant\_ iterators, const \_Value &, \_Value & >::type **reference**
- typedef \_Value **value\_type**



#### Public Member Functions

- `_Local_iterator` (const \_\_hash\_code\_base & \_\_base, [\\_Hash\\_node](#)< \_Value, \_\_cache > \* \_\_n, std::size\_t \_\_bkt, std::size\_t \_\_bkt\_count)
- reference `operator*` () const
- `_Local_iterator` & `operator++` ()
- `_Local_iterator` `operator++` (int)
- pointer `operator->` () const

##### 4.145.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __↵
constant_iterators, bool __cache>
struct std::__detail::__Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >
```

local iterators

Definition at line 1602 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.146 `std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >` Strut Template Reference

##### 4.146.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache_↵
hash_code>
struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >
```

Primary class template `_Local_iterator_base`.

Base class for local iterators, used to iterate within a bucket but not between buckets.

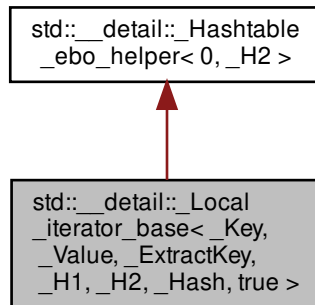
Definition at line 1150 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.147 `std::__detail::_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >` Struct Template Reference

Inheritance diagram for `std::__detail::_Local_iterator_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`:



##### Public Member Functions

- `const void * _M_curr () const`
- `std::size_t _M_get_bucket () const`

##### Protected Types

- using `__base_type` = `_Hashtable_ebo_helper<0, _H2>`
- using `__hash_code_base` = `_Hash_code_base<_Key, _Value, _ExtractKey, _H1, _H2, _Hash, true>`

##### Protected Member Functions

- `_Local_iterator_base` (`const __hash_code_base &__base, _Hash_node<_Value, true> *__p, std::size_t __bkt, std::size_t __bkt_count`)
- `void _M_incr ()`

##### Protected Attributes

- `std::size_t _M_bucket`
- `std::size_t _M_bucket_count`
- `_Hash_node<_Value, true> * _M_cur`

## 4.147.1 Detailed Description

```
template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash>
struct std::__detail::__Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >
```

Partial specialization used when nodes contain a cached hash code.

Definition at line 1423 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.148 `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_Loser` Struct Reference

## Public Attributes

- [\\_Tp \\_M\\_key](#)
- [int \\_M\\_source](#)
- [bool \\_M\\_sup](#)

## 4.148.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
```

Internal representation of a `_LoserTree` element.

Definition at line 59 of file `losertree.h`.

## 4.148.2 Member Data Documentation

4.148.2.1 `_M_key`

```
template<typename _Tp , typename _Compare >
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key
```

`_M_key` of the element in the `_LoserTree`.

Definition at line 66 of file `losertree.h`.

Referenced by `__gnu_parallel::_LoserTreeBase<_Tp, _Compare>::_insert_start()`.

#### 4.148.2.2 `_M_source`

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
```

\_\_index of the \_\_source \_\_sequence.

Definition at line 64 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

#### 4.148.2.3 `_M_sup`

```
template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup
```

flag, true iff this is a "maximum" \_\_sentinel.

Definition at line 62 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

### 4.149 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser` Struct Reference

#### Public Attributes

- `const _Tp * _M_keyp`
- `int _M_source`
- `bool _M_sup`

#### 4.149.1 Detailed Description

```
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
```

Internal representation of `_LoserTree` \_\_elements.

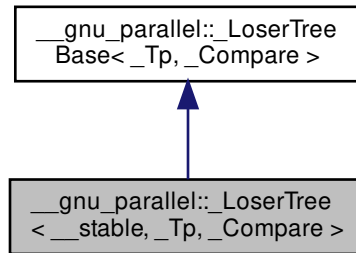
Definition at line 361 of file losertree.h.

The documentation for this struct was generated from the following file:

- [losertree.h](#)

4.150 `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >`:



## Public Member Functions

- **`_LoserTree`** (unsigned int `__k`, `_Compare` `__comp`)
- void `__delete_min_insert` (`_Tp` `__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_log_k`
- unsigned int `_M_offset`

## 4.150.1 Detailed Description

```

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >

```

Stable `_LoserTree` variant.

Provides the stable implementations of `insert_start`, `__init_winner`, `__init` and `__delete_min_insert`.

Unstable variant is done using partial specialisation below.

Definition at line 169 of file `losertree.h`.

## 4.150.2 Member Function Documentation

### 4.150.2.1 `__delete_min_insert()`

```
template<bool __stable, typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
 _Tp __key,
 bool __sup) [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.

This implementation is stable.

Definition at line 222 of file losertree.h.

### 4.150.2.2 `__get_min_source()`

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline], [inherited]
```

#### Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

### 4.150.2.3 `__insert_start()`

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline], [inherited]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

#### Parameters

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                                     |
| <code>__source</code> | <code>__index</code> of the <code>__source</code> sequence                                |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |

Definition at line 134 of file `losertree.h`.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 4.150.3 Member Data Documentation

##### 4.150.3.1 `_M_log_k`

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected], [inherited]
```

`log_2{_M_k}`

Definition at line 72 of file `losertree.h`.

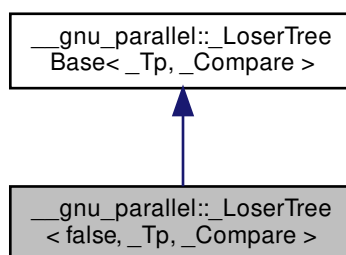
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.151 `__gnu_parallel::_LoserTree< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTree< false, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTree** (unsigned int \_\_k, \_Compare \_\_comp)
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool \_\_sup)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

### 4.151.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< false, _Tp, _Compare >
```

Unstable \_LoserTree variant.

Stability (non-stable here) is selected with partial specialization.

Definition at line 261 of file losertree.h.

### 4.151.2 Member Function Documentation

#### 4.151.2.1 \_\_delete\_min\_insert()

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
 _Tp __key,
 bool __sup) [inline]
```

Delete the \_M\_key smallest element and insert the element \_\_key instead.

#### Parameters

|              |                                                 |
|--------------|-------------------------------------------------|
| <b>__key</b> | the _M_key to insert                            |
| <b>__sup</b> | true iff __key is an explicitly marked supremum |

Definition at line 324 of file losertree.h.



## 4.151.2.2 \_\_get\_min\_source()

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline], [inherited]
```

## Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

## 4.151.2.3 \_\_init\_winner()

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner (
 unsigned int __root) [inline]
```

Computes the winner of the competition at position "`__root`".

Called recursively (starting at 0) to build the initial tree.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__root</code> | __index of the "game" to start. |
|---------------------|---------------------------------|

Definition at line 284 of file losertree.h.

## 4.151.2.4 \_\_insert\_start()

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline], [inherited]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

## Parameters

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                                     |
| <code>__source</code> | __index of the <code>__source</code> __sequence                                           |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |

Definition at line 134 of file losertree.h.

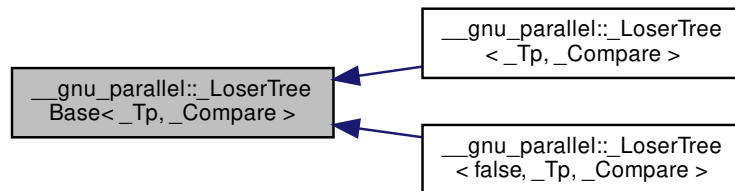
References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.152 `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >`:



#### Classes

- [struct `\_Loser`](#)

#### Public Member Functions

- [\\_LoserTreeBase](#) (unsigned int \_\_k, \_Compare \_\_comp)
- [~\\_LoserTreeBase](#) ()
- [int \\_\\_get\\_min\\_source](#) ()
- [void \\_\\_insert\\_start](#) (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)

#### Protected Attributes

- [\\_Compare `\_M\_comp`](#)
- [bool `\_M\_first\_insert`](#)
- [unsigned int `\_M\_ik`](#)
- [unsigned int `\_M\_k`](#)
- [unsigned int `\_M\_log\_k`](#)
- [\\_Loser \\* `\_M\_losers`](#)
- [unsigned int `\_M\_offset`](#)

## 4.152.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
```

Guarded loser/tournament tree.

The smallest element is at the top.

Guarding is done explicitly through one flag `_M_sup` per element, `inf` is not needed due to a better initialization routine. This is a well-performing variant.

## Parameters

|                       |                                                                      |
|-----------------------|----------------------------------------------------------------------|
| <code>_Tp</code>      | the element type                                                     |
| <code>_Compare</code> | the comparator to use, defaults to <code>std::less&lt;_Tp&gt;</code> |

Definition at line 55 of file `losertree.h`.

## 4.152.2 Constructor &amp; Destructor Documentation

## 4.152.2.1 \_LoserTreeBase()

```
template<typename _Tp , typename _Compare >
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
 unsigned int __k,
 _Compare __comp) [inline]
```

The constructor.

## Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__k</code>    | The number of sequences to merge. |
| <code>__comp</code> | The comparator to use.            |

Definition at line 94 of file `losertree.h`.

References `__gnu_parallel::_rd_log2()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

#### 4.152.2.2 ~\_LoserTreeBase()

```
template<typename _Tp , typename _Compare >
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~_LoserTreeBase () [inline]
```

The destructor.

Definition at line 118 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`.

### 4.152.3 Member Function Documentation

#### 4.152.3.1 \_\_get\_min\_source()

```
template<typename _Tp , typename _Compare >
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline]
```

##### Returns

the index of the sequence with the smallest element.

Definition at line 155 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`.

#### 4.152.3.2 \_\_insert\_start()

```
template<typename _Tp , typename _Compare >
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
 const _Tp & __key,
 int __source,
 bool __sup) [inline]
```

Initializes the sequence "`_M_source`" with the element "`__key`".

##### Parameters

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| <code>__key</code>    | the element to insert                                                                     |
| <code>__source</code> | <code>__index</code> of the <code>__source</code> sequence                                |
| <code>__sup</code>    | flag that determines whether the value to insert is an explicit <code>__supremum</code> . |

Definition at line 134 of file losertree.h.

References `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup`.

#### 4.152.4 Member Data Documentation

##### 4.152.4.1 \_M\_comp

```
template<typename _Tp , typename _Compare >
_Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp [protected]
```

`_Compare` to use.

Definition at line 78 of file losertree.h.

##### 4.152.4.2 \_M\_first\_insert

```
template<typename _Tp , typename _Compare >
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert [protected]
```

State flag that determines whether the `_LoserTree` is empty.

Only used for building the `_LoserTree`.

Definition at line 85 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

##### 4.152.4.3 \_M\_log\_k

```
template<typename _Tp , typename _Compare >
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
```

`log_2(_M_k)`

Definition at line 72 of file losertree.h.

Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`.

4.152.4.4 `_M_losers`

```
template<typename _Tp , typename _Compare >
_Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers [protected]
```

`_LoserTree` `__elements`.

Definition at line 75 of file `losertree.h`.

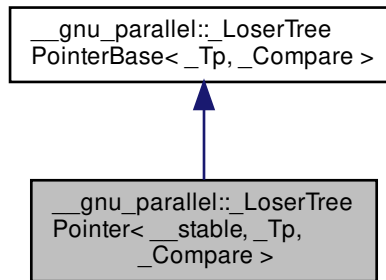
Referenced by `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_get_min_source()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_insert_start()`, `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase()`, and `__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~~LoserTreeBase()`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.153 `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >`:



## Public Member Functions

- **`_LoserTreePointer`** (unsigned int `__k`, `_Compare` `__comp=std::less<_Tp>()`)
- void **`__delete_min_insert`** (const `_Tp` & `__key`, bool `__sup`)
- int **`__get_min_source`** ()
- void **`__init`** ()
- unsigned int **`__init_winner`** (unsigned int `__root`)
- void **`__insert_start`** (const `_Tp` & `__key`, int `__source`, bool `__sup`)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 4.153.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
```

Stable `_LoserTree` implementation.

The unstable variant is implemented using partial instantiation below.

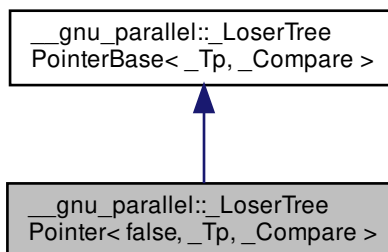
Definition at line 409 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.154 `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTreePointer` (unsigned int `__k`, `_Compare` `__comp=std::less<_Tp>()`)
- void `__delete_min_insert` (const `_Tp` &`__key`, bool `__sup`)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int `__root`)
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool `__sup`)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 4.154.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
```

Unstable `_LoserTree` implementation.

The stable variant is above.

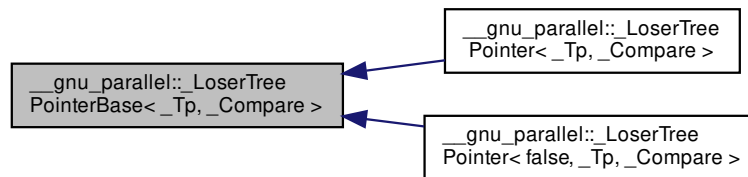
Definition at line 491 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.155 `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >`:



## Classes

- struct [\\_Loser](#)

## Public Member Functions

- **\_LoserTreePointerBase** (unsigned int \_\_k, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool \_\_sup)



## Protected Attributes

- `_Compare _M_comp`
- `unsigned int _M_ik`
- `unsigned int _M_k`
- `\_Loser * _M_losers`
- `unsigned int _M_offset`

## 4.155.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >
```

Base class of `_Loser` Tree implementation using pointers.

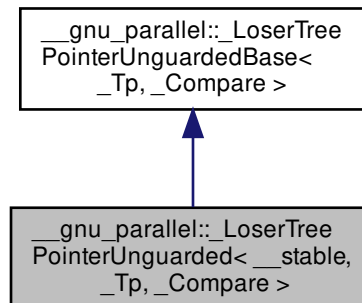
Definition at line 357 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.156 `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`:



## Public Member Functions

- `_LoserTreePointerUnguarded` (`unsigned int __k`, `const _Tp &__sentinel`, `_Compare __comp=std::less< _Tp >()`)
- `void __delete_min_insert` (`const _Tp &__key`, `bool __sup`)
- `int __get_min_source` ()
- `void __init` ()
- `unsigned int __init_winner` (`unsigned int __root`)
- `void __insert_start` (`const _Tp &__key`, `int __source`, `bool`)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 4.156.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
```

Stable unguarded `_LoserTree` variant storing pointers.

Unstable variant is implemented below using partial specialization.

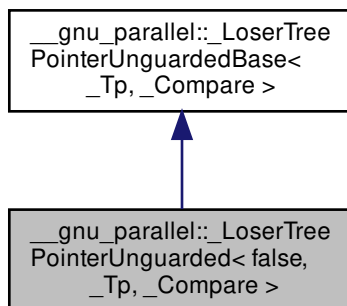
Definition at line 891 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.157 `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreePointerUnguarded** (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp`=`std::less`< `_Tp` >())
- void **\_\_delete\_min\_insert** (const `_Tp` &`__key`, bool `__sup`)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int `__root`)
- void **\_\_insert\_start** (const `_Tp` &`__key`, int `__source`, bool)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 4.157.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
```

Unstable unguarded `_LoserTree` variant storing pointers.

Stable variant is above.

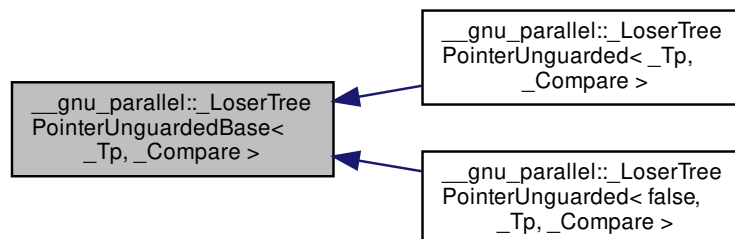
Definition at line 977 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.158 `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreePointerUnguardedBase<_Tp, _Compare>`:



## Public Member Functions

- `_LoserTreePointerUnguardedBase` (unsigned int `__k`, const `_Tp` &`__sentinel`, `_Compare` `__comp`=`std::less<_Tp>`())
- int `__get_min_source` ()
- void `__insert_start` (const `_Tp` &`__key`, int `__source`, bool)

#### Protected Attributes

- `_Compare` **`_M_comp`**
- unsigned int `_M_ik`
- unsigned int `_M_k`
- `_Loser` \* **`_M_losers`**
- unsigned int `_M_offset`

#### 4.158.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >
```

Unguarded loser tree, keeping only pointers to the elements in the tree structure.

No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.

Definition at line 828 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.159 `__gnu_parallel::_LoserTreeTraits< _Tp >` Struct Template Reference

##### Static Public Attributes

- static const bool `_M_use_pointer`

#### 4.159.1 Detailed Description

```
template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits< _Tp >
```

Traits for determining whether the loser tree should use pointers or copies.

The field "`_M_use_pointer`" is used to determine whether to use pointers in the loser trees or whether to copy the values into the loser tree.

The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.

Specialize for your data type to customize the behavior.

Example:

```
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };

template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };
```

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>_Tp</code> | type to give the loser tree traits for. |
|------------------|-----------------------------------------|

Definition at line 731 of file `multiway_merge.h`.

## 4.159.2 Member Data Documentation

4.159.2.1 `_M_use_pointer`

```
template<typename _Tp >
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer [static]
```

True iff to use pointers instead of values in loser trees.

The default behavior is to use pointers if the data type is four times as big as the pointer to it.

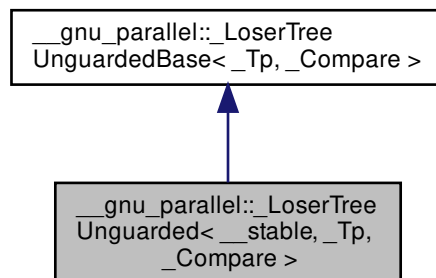
Definition at line 739 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

4.160 `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >`:



## Public Member Functions

- **\_LoserTreeUnguarded** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=std::less<\_Tp>())
- void **\_\_delete\_min\_insert** (\_Tp \_\_key, bool)
- int **\_\_get\_min\_source** ()
- void **\_\_init** ()
- unsigned int **\_\_init\_winner** (unsigned int \_\_root)
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- unsigned int **\_M\_ik**
- unsigned int **\_M\_offset**

## 4.160.1 Detailed Description

```
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
```

Stable implementation of unguarded \_LoserTree.

Unstable variant is selected below with partial specialization.

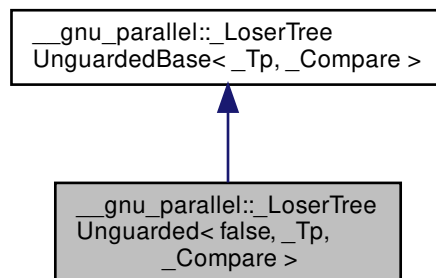
Definition at line 646 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

## 4.161 \_\_gnu\_parallel::\_LoserTreeUnguarded&lt; false, \_Tp, \_Compare &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_LoserTreeUnguarded< false, \_Tp, \_Compare >:



## Public Member Functions

- `_LoserTreeUnguarded` (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)<\_Tp>())
- void `__delete_min_insert` (\_Tp \_\_key, bool)
- int `__get_min_source` ()
- void `__init` ()
- unsigned int `__init_winner` (unsigned int \_\_root)
- void `__insert_start` (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- unsigned int `_M_ik`
- unsigned int `_M_offset`

## 4.161.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
```

Non-Stable implementation of unguarded `_LoserTree`.

Stable implementation is above.

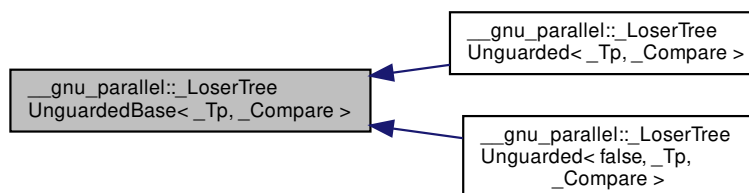
Definition at line 734 of file `losertree.h`.

The documentation for this class was generated from the following file:

- [losertree.h](#)

4.162 `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>` Class Template Reference

Inheritance diagram for `__gnu_parallel::_LoserTreeUnguardedBase<_Tp, _Compare>`:



## Public Member Functions

- **\_LoserTreeUnguardedBase** (unsigned int \_\_k, const \_Tp &\_\_sentinel, \_Compare \_\_comp=[std::less](#)< \_Tp >())
- int **\_\_get\_min\_source** ()
- void **\_\_insert\_start** (const \_Tp &\_\_key, int \_\_source, bool)

## Protected Attributes

- \_Compare **\_M\_comp**
- unsigned int **\_M\_ik**
- unsigned int **\_M\_k**
- \_Loser \* **\_M\_losers**
- unsigned int **\_M\_offset**

## 4.162.1 Detailed Description

```
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >
```

Base class for unguarded \_LoserTree implementation.

The whole element is copied into the tree structure.

No guarding is done, therefore not a single input sequence must run empty. Unused \_\_sequence heads are marked with a sentinel which is > all elements that are to be merged.

This is a very fast variant.

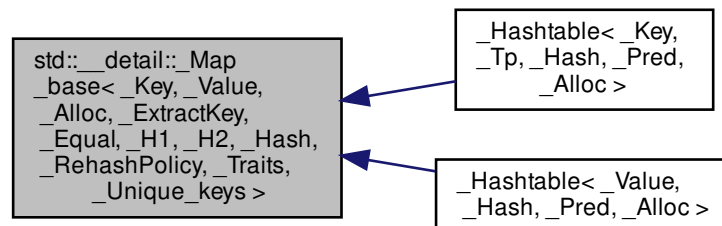
Definition at line 574 of file losertree.h.

The documentation for this class was generated from the following file:

- [losertree.h](#)

#### 4.163 std::\_\_detail::\_Map\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Map\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_↵  
RehashPolicy, \_Traits, \_Unique\_keys >:





#### 4.163.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, bool _Unique_keys = _Traits::__unique_keys::value>
struct std::__detail::_Map_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_keys >
```

Primary class template `_Map_base`.

If the hashtable has a value type of the form `pair<T1, T2>` and a key extraction policy (`_ExtractKey`) that returns the first part of the pair, the hashtable gets a `mapped_type` typedef. If it satisfies those criteria and also has unique keys, then it also gets an `operator[]`.

Definition at line 645 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.164 `std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >` Struct Template Reference

##### Public Types

- using **`mapped_type`** = typename `std::tuple_element< 1, _Pair >::type`

#### 4.164.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits>
struct std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >
```

Partial specialization, `__unique_keys` set to false.

Definition at line 651 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.165 `std::__detail::_Map_base<_Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >` Struct Template Reference

##### Public Types

- using **`iterator`** = typename `__hashtable_base::iterator`
- using **`key_type`** = typename `__hashtable_base::key_type`
- using **`mapped_type`** = typename `std::tuple_element< 1, _Pair >::type`

## Public Member Functions

- mapped\_type & **at** (const key\_type &\_\_k)
- const mapped\_type & **at** (const key\_type &\_\_k) const
- mapped\_type & **operator[]** (const key\_type &\_\_k)
- mapped\_type & **operator[]** (key\_type &&\_\_k)

### 4.165.1 Detailed Description

```
template<typename _Key, typename _Pair, typename _Alloc, typename _Equal, typename _H1, typename _H2, typename _Hash, type-
name _RehashPolicy, typename _Traits>
struct std::__detail::_Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >
```

Partial specialization, \_\_unique\_keys set to true.

Definition at line 661 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.166 std::\_\_detail::\_Mask\_range\_hashing Struct Reference

### Public Types

- typedef std::size\_t **first\_argument\_type**
- typedef std::size\_t **result\_type**
- typedef std::size\_t **second\_argument\_type**

### Public Member Functions

- result\_type **operator()** (first\_argument\_type \_\_num, second\_argument\_type \_\_den) const noexcept

### 4.166.1 Detailed Description

Range hashing function assuming that second arg is a power of 2.

Definition at line 494 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.167 `__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >` Struct Template Reference

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits< _Alloc, Metadata >::const_reference` **const\_reference**
- typedef `Metadata` **metadata\_type**

### Public Member Functions

- `const_reference` **get\_metadata** () const

### Public Attributes

- `metadata_type` **m\_metadata**

#### 4.167.1 Detailed Description

```
template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
```

Metadata base primary template.

Definition at line 67 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.168 `__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >` Struct Template Reference

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `null_type` **metadata\_type**

#### 4.168.1 Detailed Description

```
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
```

Specialization for null metadata.

Definition at line 83 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.169 `std::__detail::__Mod_range_hashing` Struct Reference

### Public Types

- typedef `std::size_t` **first\_argument\_type**
- typedef `std::size_t` **result\_type**
- typedef `std::size_t` **second\_argument\_type**

### Public Member Functions

- `result_type` **operator()** (`first_argument_type` \_\_num, `second_argument_type` \_\_den) const noexcept

#### 4.169.1 Detailed Description

Default range hashing function: use division to fold a large number into the range [0, N).

Definition at line 424 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.170 `std::_Mu<_Arg, _IsBindExp, _IsPlaceholder >` Class Template Reference

#### 4.170.1 Detailed Description

```
template<typename _Arg, bool _IsBindExp = is_bind_expression<_Arg>::value, bool _IsPlaceholder = (is_placeholder<_Arg>↵
::value > 0)>
class std::_Mu<_Arg, _IsBindExp, _IsPlaceholder >
```

Maps an argument to `bind()` into an actual argument to the bound function object `[func.bind.bind]/10`. Only the first parameter should be specified: the rest are used to determine among the various implementations. Note that, although this class is a function object, it isn't entirely normal because it takes only two parameters regardless of the number of parameters passed to the bind expression. The first parameter is the bound argument and the second parameter is a tuple containing references to the rest of the arguments.

Definition at line 288 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.171 `std::_Mu<_Arg, false, false >` Class Template Reference

### Public Member Functions

- `template<typename _CVarArg, typename _Tuple >  
constexpr _CVarArg && operator() (_CVarArg &&__arg, _Tuple &) const volatile`

#### 4.171.1 Detailed Description

```
template<typename _Arg>
class std::_Mu<_Arg, false, false >
```

If the argument is just a value, returns a reference to that value. The cv-qualifiers on the reference are determined by the caller. C++11 [func.bind.bind] p10 bullet 4.

Definition at line 372 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

## 4.172 `std::_Mu<_Arg, false, true >` Class Template Reference

### Public Member Functions

- `template<typename _Tuple >  
constexpr _Safe_tuple_element_t<(is_placeholder<_Arg >::value - 1), _Tuple > && operator() (const volatile  
_Arg &, _Tuple &__tuple) const volatile`

#### 4.172.1 Detailed Description

```
template<typename _Arg>
class std::_Mu<_Arg, false, true >
```

If the argument is a placeholder for the Nth argument, returns a reference to the Nth argument to the bind function object. C++11 [func.bind.bind] p10 bullet 3.

Definition at line 353 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

#### 4.173 `std::_Mu< _Arg, true, false >` Class Template Reference

##### Public Member Functions

- `template<typename _CVArg, typename... _Args>  
constexpr auto operator() (_CVArg &__arg, tuple< _Args... > &__tuple) const volatile -> decltype(__arg(declval< _Args >()...))`

##### 4.173.1 Detailed Description

```
template<typename _Arg>
class std::_Mu< _Arg, true, false >
```

If the argument is a bind expression, we invoke the underlying function object with the same cv-qualifiers as we are given and pass along all of our arguments (unwrapped). C++11 [func.bind.bind] p10 bullet 2.

Definition at line 317 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

#### 4.174 `std::_Mu< reference_wrapper< _Tp >, false, false >` Class Template Reference

##### Public Member Functions

- `template<typename _CVRef, typename _Tuple >  
constexpr _Tp & operator() (_CVRef &__arg, _Tuple &) const volatile`

##### 4.174.1 Detailed Description

```
template<typename _Tp>
class std::_Mu< reference_wrapper< _Tp >, false, false >
```

If the argument is `reference_wrapper<_Tp>`, returns the underlying reference. C++11 [func.bind.bind] p10 bullet 1.

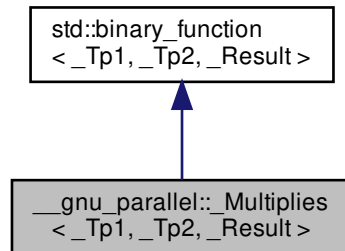
Definition at line 296 of file functional.

The documentation for this class was generated from the following file:

- [functional](#)

4.175 `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>` Struct Template Reference

Inheritance diagram for `__gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>`:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result operator()` (`const _Tp1 &__x, const _Tp2 &__y`) `const`

## 4.175.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__((*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0)))>
struct __gnu_parallel::_Multiplies<_Tp1, _Tp2, _Result>
```

Similar to `std::multiplies`, but allows two different types.

Definition at line 288 of file `base.h`.

## 4.175.2 Member Typedef Documentation

#### 4.175.2.1 first\_argument\_type

```
typedef _Tp1 std::binary_function< _Tp1 , _Tp2 , _Result >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.175.2.2 result\_type

```
typedef _Result std::binary_function< _Tp1 , _Tp2 , _Result >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.175.2.3 second\_argument\_type

```
typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

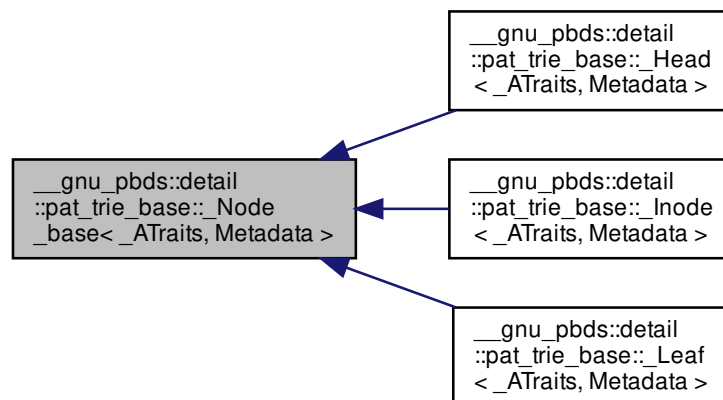
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [base.h](#)

### 4.176 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_ATraits, Metadata > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_base< \_ATraits, Metadata >:





## Public Types

- typedef `_ATraits::const_iterator` **a\_const\_iterator**
- typedef `detail::rebind_traits< _Alloc, _ATraits >::const_pointer` **a\_const\_pointer**
- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits< _Alloc, _Node_base >::pointer` **node\_pointer**
- typedef `_ATraits::type_traits` **type\_traits**

## Public Member Functions

- `_Node_base` (`node_type` type)

## Public Attributes

- `node_pointer` **m\_p\_parent**
- `const node_type` **m\_type**

### 4.176.1 Detailed Description

```
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >
```

Node base.

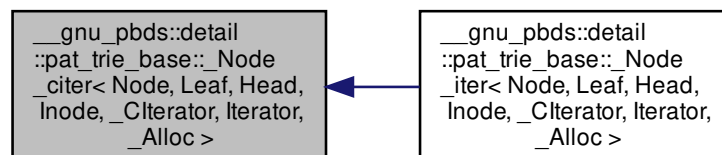
Definition at line 92 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.177 `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`:



## Public Types

- typedef value\_type **const\_reference**
- typedef trivial\_iterator\_difference\_type **difference\_type**
- typedef trivial\_iterator\_tag **iterator\_category**
- typedef rebind\_traits< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef Node::metadata\_type **metadata\_type**
- typedef value\_type **reference**
- typedef \_Alloc::size\_type **size\_type**
- typedef \_CIterator **value\_type**

## Public Member Functions

- **\_Node\_citer** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_citer get\_child** (size\_type i) const
- **metadata\_const\_reference get\_metadata** () const
- size\_type **num\_children** () const
- bool **operator!=** (const **\_Node\_citer** &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const **\_Node\_citer** &other) const
- **std::pair**< a\_const\_iterator, a\_const\_iterator > **valid\_prefix** () const

## Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

## Protected Types

- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef Node::a\_const\_pointer **a\_const\_pointer**
- typedef rebind\_traits< \_Alloc, Inode >::const\_pointer **inode\_const\_pointer**
- typedef rebind\_traits< \_Alloc, Inode >::pointer **inode\_pointer**
- typedef rebind\_traits< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef rebind\_traits< \_Alloc, Leaf >::pointer **leaf\_pointer**
- typedef rebind\_traits< \_Alloc, Node >::pointer **node\_pointer**

## 4.177.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _↔
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node const iterator.

Definition at line 810 of file pat\_trie\_base.hpp.

#### 4.177.2 Member Typedef Documentation

##### 4.177.2.1 `metadata_const_reference`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef rebind_traits<_Alloc, metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

Definition at line 862 of file `pat_trie_base.hpp`.

##### 4.177.2.2 `metadata_type`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

Definition at line 859 of file `pat_trie_base.hpp`.

#### 4.177.3 Member Function Documentation

##### 4.177.3.1 `get_child()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
_Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
 size_type i) const [inline]
```

Returns a `__const node __iterator` to the corresponding node's `i`-th child.

Definition at line 902 of file `pat_trie_base.hpp`.

#### 4.177.3.2 get\_metadata()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const [inline]
```

Metadata access.

Definition at line 885 of file pat\_trie\_base.hpp.

#### 4.177.3.3 num\_children()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const [inline]
```

Returns the number of children in the corresponding node.

Definition at line 890 of file pat\_trie\_base.hpp.

#### 4.177.3.4 operator"!="()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 918 of file pat\_trie\_base.hpp.

#### 4.177.3.5 operator\*()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _C↔
Iterator, Iterator, _Alloc >::operator* () const [inline]
```

Const access; returns the \_\_const iterator\* associated with the current leaf.

Definition at line 877 of file pat\_trie\_base.hpp.

4.177.3.6 `operator==()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

Definition at line 913 of file `pat_trie_base.hpp`.

4.177.3.7 `valid_prefix()`

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline]
```

Subtree valid prefix.

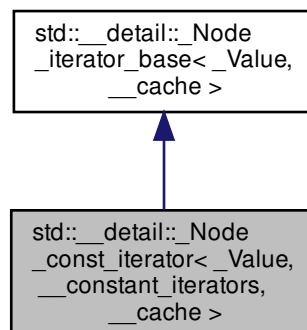
Definition at line 871 of file `pat_trie_base.hpp`.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.178 `std::__detail::_Node_const_iterator<_Value, __constant_iterators, __cache>` Struct Template Reference

Inheritance diagram for `std::__detail::_Node_const_iterator<_Value, __constant_iterators, __cache>`:



## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Value \* **pointer**
- typedef const \_Value & **reference**
- typedef \_Value **value\_type**

## Public Member Functions

- **\_Node\_const\_iterator** (\_\_node\_type \* \_\_p) noexcept
- **\_Node\_const\_iterator** (const [\\_Node\\_iterator](#)< \_Value, \_\_constant\_iterators, \_\_cache > & \_\_x) noexcept
- void **\_M\_incr** () noexcept
- reference **operator\*** () const noexcept
- [\\_Node\\_const\\_iterator](#) & **operator++** () noexcept
- [\\_Node\\_const\\_iterator](#) **operator++** (int) noexcept
- pointer **operator->** () const noexcept

## Public Attributes

- \_\_node\_type \* **\_M\_cur**

## 4.178.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_const_iterator< _Value, __constant_iterators, __cache >
```

Node const\_iterators, used to iterate through all the hashtable.

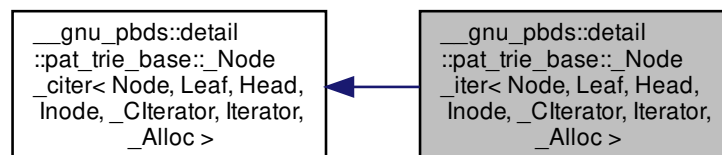
Definition at line 369 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.179 [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter](#)< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter](#)< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >:



## Public Types

- typedef value\_type **const\_reference**
- typedef [trivial\\_iterator\\_difference\\_type](#) **difference\_type**
- typedef [trivial\\_iterator\\_tag](#) **iterator\_category**
- typedef [rebind\\_traits](#)< \_Alloc, [metadata\\_type](#) >::const\_reference [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef value\_type **reference**
- typedef base\_type::size\_type **size\_type**
- typedef Iterator **value\_type**

## Public Member Functions

- **\_Node\_iter** (node\_pointer p\_nd=0, a\_const\_pointer p\_traits=0)
- **\_Node\_iter get\_child** (size\_type i) const
- [metadata\\_const\\_reference get\\_metadata](#) () const
- size\_type [num\\_children](#) () const
- bool [operator!=](#) (const [\\_Node\\_citer](#) &other) const
- reference [operator\\*](#) () const
- bool [operator==](#) (const [\\_Node\\_citer](#) &other) const
- [std::pair](#)< a\_const\_iterator, a\_const\_iterator > [valid\\_prefix](#) () const

## Public Attributes

- node\_pointer **m\_p\_nd**
- a\_const\_pointer **m\_p\_traits**

## Protected Types

- typedef Node::a\_const\_iterator **a\_const\_iterator**
- typedef [rebind\\_traits](#)< \_Alloc, Inode >::const\_pointer **inode\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::const\_pointer **leaf\_const\_pointer**
- typedef [rebind\\_traits](#)< \_Alloc, Leaf >::pointer **leaf\_pointer**

### 4.179.1 Detailed Description

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _↵
Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 934 of file `pat_trie_base.hpp`.

## 4.179.2 Member Typedef Documentation

### 4.179.2.1 metadata\_const\_reference

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef rebind_traits<_Alloc, metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference [inherited]
```

Const metadata reference type.

Definition at line 862 of file `pat_trie_base.hpp`.

### 4.179.2.2 metadata\_type

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type [inherited]
```

Metadata type.

Definition at line 859 of file `pat_trie_base.hpp`.

## 4.179.3 Member Function Documentation

### 4.179.3.1 get\_child()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
_Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
 size_type i) const [inline]
```

Returns a node `__iterator` to the corresponding node's i-th child.

Definition at line 966 of file `pat_trie_base.hpp`.



#### 4.179.3.2 get\_metadata()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const [inline], [inherited]
```

Metadata access.

Definition at line 885 of file pat\_trie\_base.hpp.

#### 4.179.3.3 num\_children()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const [inline], [inherited]
```

Returns the number of children in the corresponding node.

Definition at line 890 of file pat\_trie\_base.hpp.

#### 4.179.3.4 operator!=(())

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!=((
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 918 of file pat\_trie\_base.hpp.

#### 4.179.3.5 operator\*()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* () const [inline]
```

Access; returns the iterator\* associated with the current leaf.

Definition at line 958 of file pat\_trie\_base.hpp.

#### 4.179.3.6 operator==()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
 const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 913 of file pat\_trie\_base.hpp.

#### 4.179.3.7 valid\_prefix()

```
template<typename Node , typename Leaf , typename Head , typename Inode , typename _CIterator ,
typename Iterator , typename _Alloc >
std::pair<a_const_iterator, a_const_iterator> __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const [inline], [inherited]
```

Subtree valid prefix.

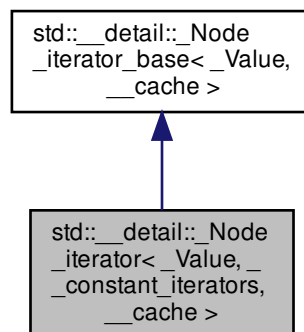
Definition at line 871 of file pat\_trie\_base.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

### 4.180 std::\_\_detail::\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache > Struct Template Reference

Inheritance diagram for std::\_\_detail::\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >:



## Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- using **pointer** = typename `std::conditional< __constant_iterators, const _Value *, _Value * >::type`
- using **reference** = typename `std::conditional< __constant_iterators, const _Value &, _Value & >::type`
- typedef `_Value` **value\_type**

## Public Member Functions

- **\_Node\_iterator** (`__node_type * __p`) noexcept
- void **\_M\_incr** () noexcept
- reference **operator\*** () const noexcept
- **\_Node\_iterator** & **operator++** () noexcept
- **\_Node\_iterator** **operator++** (int) noexcept
- pointer **operator->** () const noexcept

## Public Attributes

- `__node_type * _M_cur`

## 4.180.1 Detailed Description

```
template<typename _Value, bool __constant_iterators, bool __cache>
struct std::__detail::_Node_iterator< _Value, __constant_iterators, __cache >
```

Node iterators, used to iterate through all the hashtable.

Definition at line 318 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

4.181 `std::__detail::_Node_iterator_base< _Value, _Cache_hash_code >` Struct Template Reference

## Public Types

- using `__node_type` = `_Hash_node< _Value, _Cache_hash_code >`

## Public Member Functions

- **\_Node\_iterator\_base** (`__node_type * __p`) noexcept
- void **\_M\_incr** () noexcept

#### Public Attributes

- [\\_\\_node\\_type](#) \* [\\_M\\_cur](#)

#### 4.181.1 Detailed Description

```
template<typename _Value, bool _Cache_hash_code>
struct std::__detail::__Node_iterator_base< _Value, _Cache_hash_code >
```

Base class for node iterators.

Definition at line 288 of file hashtable\_policy.h.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 4.182 [\\_\\_gnu\\_debug::\\_\\_Not\\_equal\\_to](#)< [\\_Type](#) > Class Template Reference

#### Public Member Functions

- [\\_Not\\_equal\\_to](#) (const [\\_Type](#) &\_\_v)
- bool **operator()** (const [\\_Type](#) &\_\_x) const

#### 4.182.1 Detailed Description

```
template<typename _Type>
class __gnu_debug::__Not_equal_to< _Type >
```

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both `bind1st` and `not_equal`.

Definition at line 44 of file `safe_sequence.h`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

### 4.183 [std::\\_\\_Not\\_fn](#)< [\\_Fn](#) > Class Template Reference

#### Public Member Functions

- `template<typename _Fn2 >`  
`constexpr \_Not\_fn (\_Fn2 &&__fn, int)`
- [\\_Not\\_fn](#) (const [\\_Not\\_fn](#) &\_\_fn)=default
- [\\_Not\\_fn](#) ([\\_Not\\_fn](#) &&\_\_fn)=default

## Public Attributes

- `template<typename... _Args>`  
`decltype(_S_not< __inv_res_t< _Fn &, _Args... >>()) constexpr operator() (_Args &&... __args) &noexcept(←`  
`__is_nothrow_invocable< _Fn &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &, _Args... >>()))`
- `template<typename... _Args>`  
`decltype(_S_not< __inv_res_t< _Fn const &, _Args... >>()) constexpr operator() (_Args &&... __args) const`  
`&&noexcept(__is_nothrow_invocable< _Fn const &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`const &, _Args... >>()))`
- `template<typename... _Args>`  
`decltype(_S_not< __inv_res_t< _Fn &&, _Args... >>()) constexpr operator() (_Args &&... __args)`  
`&&noexcept(__is_nothrow_invocable< _Fn &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`&&, _Args... >>()))`
- `template<typename... _Args>`  
`decltype(_S_not< __inv_res_t< _Fn const &&, _Args... >>()) constexpr operator() (_Args &&... __args) const`  
`&&noexcept(__is_nothrow_invocable< _Fn const &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn`  
`const &&, _Args... >>()))`

## 4.183.1 Detailed Description

```
template<typename _Fn>
class std::_Not_fn< _Fn >
```

Generalized negator.

Definition at line 919 of file `functional`.

The documentation for this class was generated from the following file:

- [functional](#)

4.184 `__gnu_parallel::_Nothing` Struct Reference

## Public Member Functions

- `template<typename _It >`  
`void operator() (_It __i)`

## 4.184.1 Detailed Description

Functor doing nothing.

For some `__reduction` tasks (this is not a function object, but is passed as `__selector` `__dummy` parameter.

Definition at line 288 of file `for_each_selectors.h`.





## 4.184.2 Member Function Documentation

4.184.2.1 `operator()`

```
template<typename _It >
void __gnu_parallel::_Nothing::operator() (
 _It __i) [inline]
```

Functor execution.

**Parameters**

|                                                                                   |                              |
|-----------------------------------------------------------------------------------|------------------------------|
|  | iterator referencing object. |
|  |                              |
|  |                              |
|  |                              |

Definition at line 294 of file `for_each_selectors.h`.

The documentation for this struct was generated from the following file:

- [for\\_each\\_selectors.h](#)

#### 4.185 `__gnu_parallel::_Piece<_DifferenceTp >` Struct Template Reference

**Public Types**

- `typedef _DifferenceTp DifferenceType`

**Public Attributes**

- `_DifferenceType \_M\_begin`
- `_DifferenceType \_M\_end`

##### 4.185.1 Detailed Description

```
template<typename _DifferenceTp>
struct __gnu_parallel::_Piece<_DifferenceTp >
```

Subsequence description.

Definition at line 46 of file `multiway_mergesort.h`.

##### 4.185.2 Member Data Documentation

###### 4.185.2.1 `_M_begin`

```
template<typename _DifferenceTp >
_DifferenceType __gnu_parallel::_Piece<_DifferenceTp >::_M_begin
```

Begin of subsequence.

Definition at line 51 of file `multiway_mergesort.h`.

## 4.185.2.2 \_M\_end

```
template<typename _DifferenceTp >
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
```

End of subsequence.

Definition at line 54 of file multiway\_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 4.186 std::\_Placeholder&lt;\_Num&gt; Struct Template Reference

## 4.186.1 Detailed Description

```
template<int _Num>
struct std::_Placeholder<_Num>
```

The type of placeholder objects defined by libstdc++.

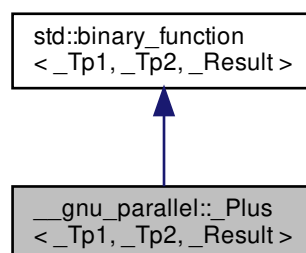
Definition at line 209 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

## 4.187 \_\_gnu\_parallel::\_Plus&lt;\_Tp1, \_Tp2, \_Result&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_parallel::\_Plus<\_Tp1, \_Tp2, \_Result>:



## Public Types

- typedef `_Tp1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Tp2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Result` **operator()** (const `_Tp1` &\_\_x, const `_Tp2` &\_\_y) const

### 4.187.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Result = __typeof__ (*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>
struct __gnu_parallel::Plus<_Tp1, _Tp2, _Result >
```

Similar to `std::plus`, but allows two different types.

Definition at line 272 of file `base.h`.

### 4.187.2 Member Typedef Documentation

#### 4.187.2.1 first\_argument\_type

```
typedef _Tp1 std::binary_function<_Tp1 , _Tp2 , _Result >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 4.187.2.2 result\_type

```
typedef _Result std::binary_function<_Tp1 , _Tp2 , _Result >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.



4.187.2.3 `second_argument_type`

```
typedef _Tp2 std::binary_function< _Tp1 , _Tp2 , _Result >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [base.h](#)

4.188 `__gnu_parallel::_PMWSSortingData<_RAIter>` Struct Template Reference

## Public Types

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef [std::iterator\\_traits](#)< `_RAIter` > **`_TraitsType`**
- typedef `_TraitsType::value_type` **`_ValueType`**

## Public Attributes

- [\\_ThreadIndex](#) `_M_num_threads`
- `_DifferenceType` \* [\\_M\\_offsets](#)
- [std::vector](#)< [\\_Piece](#)< `_DifferenceType` > > \* [\\_M\\_pieces](#)
- `_ValueType` \* [\\_M\\_samples](#)
- `_RAIter` [\\_M\\_source](#)
- `_DifferenceType` \* [\\_M\\_starts](#)
- `_ValueType` \*\* [\\_M\\_temporary](#)

## 4.188.1 Detailed Description

```
template<typename _RAIter>
struct __gnu_parallel::_PMWSSortingData<_RAIter>
```

Data accessed by all threads.

PMWMS = parallel multiway mergesort

Definition at line 61 of file `multiway_mergesort.h`.

## 4.188.2 Member Data Documentation

#### 4.188.2.1 `_M_num_threads`

```
template<typename _RAIter>
__ThreadIndex __gnu_parallel::_PMWMSortingData< _RAIter >::_M_num_threads
```

Number of threads involved.

Definition at line 68 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

#### 4.188.2.2 `_M_offsets`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_offsets
```

Offsets to add to the found positions.

Definition at line 83 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

#### 4.188.2.3 `_M_pieces`

```
template<typename _RAIter>
std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_pieces
```

Pieces of data to merge [thread][\_\_sequence].

Definition at line 86 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`.

#### 4.188.2.4 `_M_samples`

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWMSortingData< _RAIter >::_M_samples
```

Samples.

Definition at line 80 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::__determine_samples()`, and `__gnu_parallel::parallel_sort_mwms()`.

4.188.2.5 `_M_source`

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWSSortingData<_RAIter>::_M_source
```

Input `__begin`.

Definition at line 71 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::_determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.188.2.6 `_M_starts`

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWSSortingData<_RAIter>::_M_starts
```

Start indices, per thread.

Definition at line 74 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::_determine_samples()`, `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

4.188.2.7 `_M_temporary`

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWSSortingData<_RAIter>::_M_temporary
```

Storage in which to sort.

Definition at line 77 of file `multiway_mergesort.h`.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

4.189 `__gnu_cxx::Pointer_adapter<_Storage_policy>` Class Template Reference

Inherits `_Storage_policy`.

## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Storage\_policy::element\_type **element\_type**
- typedef std::random\_access\_iterator\_tag **iterator\_category**
- typedef \_Pointer\_adapter **pointer**
- typedef \_Reference\_type< element\_type >::reference **reference**
- typedef \_Unqualified\_type< element\_type >::type **value\_type**

## Public Member Functions

- **\_Pointer\_adapter** (element\_type \* \_\_arg=0)
- **\_Pointer\_adapter** (const \_Pointer\_adapter & \_\_arg)
- template<typename \_Up >  
  **\_Pointer\_adapter** (\_Up \* \_\_arg)
- template<typename \_Up >  
  **\_Pointer\_adapter** (const \_Pointer\_adapter< \_Up > & \_\_arg)
- **operator bool** () const
- reference **operator\*** () const
- **\_Pointer\_adapter** & **operator++** ()
- **\_Pointer\_adapter** **operator++** (int)
- **\_Pointer\_adapter** & **operator+=** (short \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (unsigned short \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (int \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (unsigned int \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (long \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (unsigned long \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (long long \_\_offset)
- **\_Pointer\_adapter** & **operator+=** (unsigned long long \_\_offset)
- template<typename \_Up >  
  std::ptrdiff\_t **operator-** (const \_Pointer\_adapter< \_Up > & \_\_rhs) const
- **\_Pointer\_adapter** & **operator--** ()
- **\_Pointer\_adapter** **operator--** (int)
- **\_Pointer\_adapter** & **operator-=** (short \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (unsigned short \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (int \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (unsigned int \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (long \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (unsigned long \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (long long \_\_offset)
- **\_Pointer\_adapter** & **operator-=** (unsigned long long \_\_offset)
- element\_type \* **operator->** () const
- **\_Pointer\_adapter** & **operator=** (const \_Pointer\_adapter & \_\_arg)
- template<typename \_Up >  
  **\_Pointer\_adapter** & **operator=** (const \_Pointer\_adapter< \_Up > & \_\_arg)
- template<typename \_Up >  
  **\_Pointer\_adapter** & **operator=** (\_Up \* \_\_arg)
- reference **operator[]** (std::ptrdiff\_t \_\_index) const

## Friends

- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, short \_\_offset)
- `_Pointer_adapter operator+` (short \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, unsigned short \_\_offset)
- `_Pointer_adapter operator+` (unsigned short \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, int \_\_offset)
- `_Pointer_adapter operator+` (int \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, unsigned int \_\_offset)
- `_Pointer_adapter operator+` (unsigned int \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, long \_\_offset)
- `_Pointer_adapter operator+` (long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, unsigned long \_\_offset)
- `_Pointer_adapter operator+` (unsigned long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, long long \_\_offset)
- `_Pointer_adapter operator+` (long long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (unsigned long long \_\_offset, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator+` (const `_Pointer_adapter` &\_\_lhs, unsigned long long \_\_offset)
- `std::ptrdiff_t operator-` (const `_Pointer_adapter` &\_\_lhs, `element_type` \* \_\_rhs)
- `std::ptrdiff_t operator-` (`element_type` \* \_\_lhs, const `_Pointer_adapter` &\_\_rhs)
- `template<typename _Up>`  
`std::ptrdiff_t operator-` (const `_Pointer_adapter` &\_\_lhs, `_Up` \* \_\_rhs)
- `template<typename _Up>`  
`std::ptrdiff_t operator-` (`_Up` \* \_\_lhs, const `_Pointer_adapter` &\_\_rhs)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, short \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned short \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, int \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned int \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, long \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned long \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, long long \_\_offset)
- `_Pointer_adapter operator-` (const `_Pointer_adapter` &\_\_lhs, unsigned long long \_\_offset)

## 4.189.1 Detailed Description

```
template<typename _Storage_policy>
class __gnu_cxx::Pointer_adapter<_Storage_policy>
```

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.

The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' `Allocator::pointer` is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use

the Allocator::pointer typedef appropriately for pointer types. 2) if you need pointer casting, use the `__pointer_cast<>` functions from `ext/cast.h`. This allows pointer cast operations to be overloaded as necessary by custom pointers.

Note: The const qualifier works with this pointer adapter as follows:

```
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const
_Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_↵
adapter<_Std_pointer_impl<const _Tp> >;
```

Definition at line 284 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 4.190 `std::__detail::_Power2_rehash_policy` Struct Reference

### Public Types

- using `__has_load_factor` = [true\\_type](#)
- typedef `std::size_t` `_State`

### Public Member Functions

- `_Power2_rehash_policy` (float `__z`=1.0) noexcept
- `std::size_t` `_M_bkt_for_elements` (std::size\_t `__n`) const noexcept
- [std::pair](#)< bool, std::size\_t > `_M_need_rehash` (std::size\_t `__n_bkt`, std::size\_t `__n_elt`, std::size\_t `__n_ins`) noexcept
- `std::size_t` `_M_next_bkt` (std::size\_t `__n`) noexcept
- void `_M_reset` () noexcept
- void `_M_reset` (\_State `__state`) noexcept
- \_State `_M_state` () const noexcept
- float `max_load_factor` () const noexcept

### Public Attributes

- float `_M_max_load_factor`
- `std::size_t` `_M_next_resize`

### Static Public Attributes

- static const std::size\_t `_S_growth_factor`

#### 4.190.1 Detailed Description

Rehash policy providing power of 2 bucket numbers. Avoids modulo operations.

Definition at line 522 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

### 4.191 `std::__detail::_Prime_rehash_policy` Struct Reference

#### Public Types

- using `__has_load_factor` = `true_type`
- typedef `std::size_t` `_State`

#### Public Member Functions

- `_Prime_rehash_policy` (`float __z=1.0`) `noexcept`
- `std::size_t` `_M_bkt_for_elements` (`std::size_t __n`) `const`
- `std::pair`< `bool`, `std::size_t` > `_M_need_rehash` (`std::size_t __n_bkt`, `std::size_t __n_elt`, `std::size_t __n_ins`) `const`
- `std::size_t` `_M_next_bkt` (`std::size_t __n`) `const`
- `void` `_M_reset` () `noexcept`
- `void` `_M_reset` (`_State __state`)
- `_State` `_M_state` () `const`
- `float` `max_load_factor` () `const` `noexcept`

#### Public Attributes

- `float` `_M_max_load_factor`
- `std::size_t` `_M_next_resize`

#### Static Public Attributes

- static `const` `std::size_t` `_S_growth_factor`

#### 4.191.1 Detailed Description

Default value for rehash policy. Bucket size is (usually) the smallest prime that keeps the load factor small enough.

Definition at line 445 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

## 4.192 `__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>` Class Template Reference

### Public Types

- typedef `_DifferenceTp` **`DifferenceType`**
- typedef `_PseudoSequenceIterator<_Tp, uint64_t>` **`iterator`**

### Public Member Functions

- `_PseudoSequence` (const `_Tp` & `__val`, `_DifferenceType` `__count`)
- `iterator begin` () const
- `iterator end` () const

#### 4.192.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>
```

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

#### Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M_value</code> type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

Definition at line 359 of file `base.h`.

#### 4.192.2 Constructor & Destructor Documentation

##### 4.192.2.1 `_PseudoSequence()`

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::_PseudoSequence (
 const _Tp & __val,
 _DifferenceType __count) [inline]
```

Constructor.

#### Parameters

|                      |                             |
|----------------------|-----------------------------|
| <code>__val</code>   | Element of the sequence.    |
| <code>__count</code> | Number of (virtual) copies. |



Definition at line 371 of file base.h.

#### 4.192.3 Member Function Documentation

##### 4.192.3.1 `begin()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::begin () const [inline]
```

Begin iterator.

Definition at line 376 of file base.h.

##### 4.192.3.2 `end()`

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence<_Tp, _DifferenceTp>::end () const [inline]
```

End iterator.

Definition at line 381 of file base.h.

The documentation for this class was generated from the following file:

- [base.h](#)

#### 4.193 `__gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>` Class Template Reference

##### Public Types

- typedef `_DifferenceTp` **`_DifferenceType`**

##### Public Member Functions

- **`_PseudoSequenceliterator`** (const `_Tp` &\_\_val, `_DifferenceType` \_\_pos)
- bool **`operator!=`** (const [\\_PseudoSequenceliterator](#) &\_\_i2)
- const `_Tp` & **`operator*`** () const
- [\\_PseudoSequenceliterator](#) & **`operator++`** ()
- [\\_PseudoSequenceliterator](#) **`operator++`** (int)
- `_DifferenceType` **`operator-`** (const [\\_PseudoSequenceliterator](#) &\_\_i2)
- bool **`operator==`** (const [\\_PseudoSequenceliterator](#) &\_\_i2)
- const `_Tp` & **`operator[]`** (`_DifferenceType`) const

##### 4.193.1 Detailed Description

```
template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequenceliterator<_Tp, _DifferenceTp>
```

`_Iterator` associated with `__gnu_parallel::_PseudoSequence`. It features the usual random-access iterator functionality.

**Parameters**

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_Tp</code>           | Sequence <code>_M_value</code> type. |
| <code>_DifferenceTp</code> | Sequence difference type.            |

Definition at line 306 of file `base.h`.

The documentation for this class was generated from the following file:

- [base.h](#)

**4.194 `__gnu_parallel::__QSBThreadLocal<_RAIter>` Struct Template Reference****Public Types**

- typedef `_TraitsType::difference_type` **`_DifferenceType`**
- typedef `std::pair<_RAIter, _RAIter>` [\\_Piece](#)
- typedef `std::iterator_traits<_RAIter>` **`_TraitsType`**

**Public Member Functions**

- [\\_QSBThreadLocal](#) (int `__queue_size`)

**Public Attributes**

- volatile `_DifferenceType *` [\\_M\\_elements\\_leftover](#)
- [\\_Piece](#) [\\_M\\_global](#)
- [\\_Piece](#) [\\_M\\_initial](#)
- `_RestrictedBoundedConcurrentQueue<_Piece>` [\\_M\\_leftover\\_parts](#)
- `_ThreadIndex` [\\_M\\_num\\_threads](#)

**4.194.1 Detailed Description**

```
template<typename _RAIter>
struct __gnu_parallel::__QSBThreadLocal<_RAIter>
```

Information local to one thread in the parallel quicksort run.

Definition at line 65 of file `balanced_quicksort.h`.

**4.194.2 Member Typedef Documentation**

#### 4.194.2.1 \_Piece

```
template<typename _RAIter>
typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal<_RAIter>::_Piece
```

Continuous part of the sequence, described by an iterator pair.

Definition at line 72 of file balanced\_quicksort.h.

### 4.194.3 Constructor & Destructor Documentation

#### 4.194.3.1 \_QSBThreadLocal()

```
template<typename _RAIter>
__gnu_parallel::_QSBThreadLocal<_RAIter>::_QSBThreadLocal (
 int __queue_size) [inline]
```

Constructor.

##### Parameters

|                           |                                  |
|---------------------------|----------------------------------|
| <code>__queue_size</code> | size of the work-stealing queue. |
|---------------------------|----------------------------------|

Definition at line 91 of file balanced\_quicksort.h.

### 4.194.4 Member Data Documentation

#### 4.194.4.1 \_M\_elements\_leftover

```
template<typename _RAIter>
volatile _DifferenceType* __gnu_parallel::_QSBThreadLocal<_RAIter>::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.

Definition at line 84 of file balanced\_quicksort.h.

Referenced by `__gnu_parallel::__parallel_sort_qsb()`.

#### 4.194.4.2 `_M_global`

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

Definition at line 87 of file `balanced_quicksort.h`.

#### 4.194.4.3 `_M_initial`

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.

Definition at line 75 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_qsb_conquer()`, and `__gnu_parallel::_qsb_local_sort_with_helping()`.

#### 4.194.4.4 `_M_leftover_parts`

```
template<typename _RAIter>
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_↵
leftover_parts
```

Work-stealing queue.

Definition at line 78 of file `balanced_quicksort.h`.

#### 4.194.4.5 `_M_num_threads`

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads
```

Number of threads involved in this algorithm.

Definition at line 81 of file `balanced_quicksort.h`.

Referenced by `__gnu_parallel::_qsb_local_sort_with_helping()`.

The documentation for this struct was generated from the following file:

- [balanced\\_quicksort.h](#)

## 4.195 `std::__detail::_Quoted_string<_String, _CharT>` Struct Template Reference

### Public Member Functions

- **`_Quoted_string`** (`_String __str, _CharT __del, _CharT __esc`)
- `_Quoted_string` & **`operator=`** (`_Quoted_string` &)=delete

### Public Attributes

- `_CharT` **`_M_delim`**
- `_CharT` **`_M_escape`**
- `_String` **`_M_string`**

#### 4.195.1 Detailed Description

```
template<typename _String, typename _CharT>
struct std::__detail::_Quoted_string<_String, _CharT>
```

Struct for delimited strings.

Definition at line 49 of file `quoted_string.h`.

The documentation for this struct was generated from the following file:

- [quoted\\_string.h](#)

## 4.196 `__gnu_parallel::_RandomNumber` Class Reference

### Public Member Functions

- `_RandomNumber` ()
- `_RandomNumber` (`uint32_t __seed, uint64_t _M_supremum=0x100000000ULL`)
- unsigned long `__genrand_bits` (`int __bits`)
- `uint32_t` `operator()` ()
- `uint32_t` `operator()` (`uint64_t local_supremum`)

#### 4.196.1 Detailed Description

Random number generator, based on the Mersenne twister.

Definition at line 42 of file `random_number.h`.

#### 4.196.2 Constructor & Destructor Documentation

**4.196.2.1** `_RandomNumber()` [1/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber () [inline]
```

Default constructor. Seed with 0.

Definition at line 74 of file random\_number.h.

**4.196.2.2** `_RandomNumber()` [2/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber (
 uint32_t __seed,
 uint64_t _M_supremum = 0x100000000ULL) [inline]
```

Constructor.

**Parameters**

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| <code>__seed</code>      | Random <code>__seed</code> .                                                    |
| <code>_M_supremum</code> | Generate integer random numbers in the interval <code>[0, _M_supremum)</code> . |

Definition at line 85 of file random\_number.h.

**4.196.3** **Member Function Documentation****4.196.3.1** `__genrand_bits()`

```
unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (
 int __bits) [inline]
```

Generate a number of random bits, run-time parameter.

**Parameters**

|                     |                             |
|---------------------|-----------------------------|
| <code>__bits</code> | Number of bits to generate. |
|---------------------|-----------------------------|

Definition at line 109 of file random\_number.h.

**4.196.3.2** `operator()()` [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() () [inline]
```

Generate unsigned random 32-bit integer.

Definition at line 94 of file `random_number.h`.

#### 4.196.3.3 `operator()` [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
 uint64_t local_supremum) [inline]
```

Generate unsigned random 32-bit integer in the interval [0,local\_supremum).

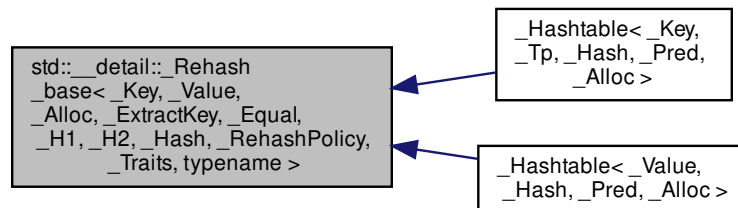
Definition at line 100 of file `random_number.h`.

The documentation for this class was generated from the following file:

- [random\\_number.h](#)

#### 4.197 `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename >` Struct Template Reference

Inheritance diagram for `std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename >`:



#### 4.197.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits, typename = __detected_or_t<false_type, __has_load_factor, _RehashPolicy>>
```

```
struct std::__detail::_Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typename >
```

Primary class template `_Rehash_base`.

Give hashtable the `max_load_factor` functions and reserve iff the rehash policy supports it.

Definition at line 1049 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.198 `std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false_type >` Struct Template Reference

##### 4.198.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false_type >
```

Specialization when rehash policy doesn't provide load factor management.

Definition at line 1056 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)

#### 4.199 `std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true_type >` Struct Template Reference

##### Public Types

- using `__hashtable` = `_Hashtable`<\_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

##### Public Member Functions

- float `max_load_factor` () const noexcept
- void `max_load_factor` (float \_\_z)
- void `reserve` (std::size\_t \_\_n)

##### 4.199.1 Detailed Description

```
template<typename _Key, typename _Value, typename _Alloc, typename _ExtractKey, typename _Equal, typename _H1, typename _H2,
typename _Hash, typename _RehashPolicy, typename _Traits>
struct std::__detail::_Rehash_base<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true_type >
```

Specialization when rehash policy provide load factor management.

Definition at line 1067 of file `hashtable_policy.h`.

The documentation for this struct was generated from the following file:

- [hashtable\\_policy.h](#)



## 4.200 `__gnu_cxx::_Relative_pointer_impl<_Tp>` Class Template Reference

### Public Types

- typedef `_Tp` **element\_type**

### Public Member Functions

- `_Tp * get () const`
- bool **operator**< (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- bool **operator**== (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- void **set** (`_Tp *__arg`)

#### 4.200.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which stores the pointer's address as an offset value which is relative to its own address.

This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.

Definition at line 112 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

## 4.201 `__gnu_cxx::_Relative_pointer_impl<const _Tp>` Class Template Reference

### Public Types

- typedef const `_Tp` **element\_type**

### Public Member Functions

- const `_Tp * get () const`
- bool **operator**< (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- bool **operator**== (const [\\_Relative\\_pointer\\_impl](#) &\_\_rarg) const
- void **set** (const `_Tp *__arg`)

#### 4.201.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const _Tp >
```

`Relative_pointer_impl` needs a specialization for `const T` because of the casting done during pointer arithmetic.

Definition at line 164 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

#### 4.202 \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp > Class Template Reference

##### Public Member Functions

- [\\_RestrictedBoundedConcurrentQueue](#) ([\\_SequenceIndex](#) \_\_max\_size)
- [~\\_RestrictedBoundedConcurrentQueue](#) ()
- [bool pop\\_back](#) ([\\_Tp](#) &\_\_t)
- [bool pop\\_front](#) ([\\_Tp](#) &\_\_t)
- [void push\\_front](#) ([const \\_Tp](#) &\_\_t)

#### 4.202.1 Detailed Description

```
template<typename _Tp>
class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >
```

Double-ended queue of bounded size, allowing lock-free atomic access. `push_front()` and `pop_front()` must not be called concurrently to each other, while `pop_back()` can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>_Tp</code> | Contained element type. |
|------------------|-------------------------|

Definition at line 52 of file `queue.h`.

#### 4.202.2 Constructor & Destructor Documentation

## 4.202.2.1 \_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue (
 __SequenceIndex __max_size) [inline]
```

Constructor. Not to be called concurrent, of course.

## Parameters

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__max_size</code> | Maximal number of elements to be contained. |
|-------------------------|---------------------------------------------|

Definition at line 68 of file queue.h.

## 4.202.2.2 ~\_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~~_RestrictedBoundedConcurrentQueue ()
[inline]
```

Destructor. Not to be called concurrent, of course.

Definition at line 77 of file queue.h.

## 4.202.3 Member Function Documentation

## 4.202.3.1 pop\_back()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (
 _Tp & __t) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop\_front().

Definition at line 127 of file queue.h.

## 4.202.3.2 pop\_front()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
 _Tp & __t) [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop\_front().

Definition at line 100 of file queue.h.

#### 4.202.3.3 push\_front()

```
template<typename _Tp>
void __gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >::push_front (
 const _Tp & __t) [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with pop\_front().

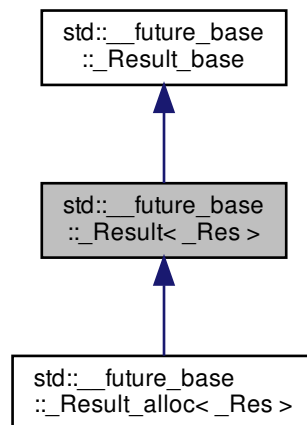
Definition at line 83 of file queue.h.

The documentation for this class was generated from the following file:

- [queue.h](#)

#### 4.203 std::\_\_future\_base::\_Result< \_Res > Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_Result< \_Res >:



##### Public Types

- typedef `_Res` **result\_type**

##### Public Member Functions

- void **\_M\_set** (const `_Res` &\_\_res)
- void **\_M\_set** (`_Res` &&\_\_res)
- `_Res` & **\_M\_value** () noexcept

## Public Attributes

- [exception\\_ptr](#) **M\_error**

## 4.203.1 Detailed Description

```
template<typename _Res>
struct std::__future_base::_Result< _Res >
```

A result object that has storage for an object of type `_Res`.

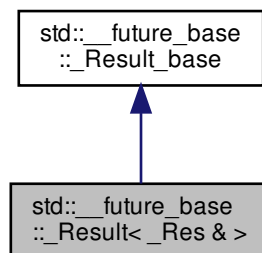
Definition at line 227 of file `future`.

The documentation for this struct was generated from the following file:

- [future](#)

## 4.204 std::\_\_future\_base::\_Result&lt; \_Res &gt; Struct Template Reference

Inheritance diagram for `std::__future_base::_Result< _Res >`:



## Public Types

- `typedef _Res & result_type`

## Public Member Functions

- `_Res & M_get () noexcept`
- `void M_set (_Res &__res) noexcept`

#### Public Attributes

- [exception\\_ptr](#) **\_M\_error**

#### 4.204.1 Detailed Description

```
template<typename _Res>
struct std::__future_base::_Result< _Res & >
```

Partial specialization for reference types.

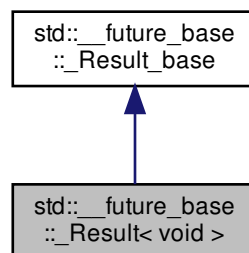
Definition at line 638 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

#### 4.205 std::\_\_future\_base::\_Result< void > Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_Result< void >:



#### Public Types

- typedef void **result\_type**

#### Public Attributes

- [exception\\_ptr](#) **\_M\_error**

## 4.205.1 Detailed Description

```
template<>
struct std::__future_base::_Result< void >
```

Explicit specialization for void.

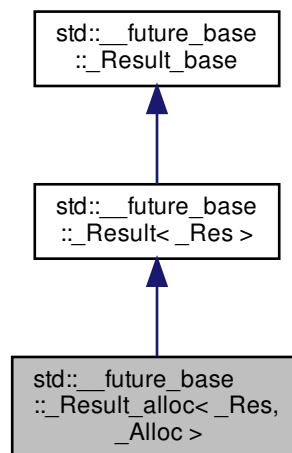
Definition at line 658 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

## 4.206 std::\_\_future\_base::\_Result\_alloc&lt; \_Res, \_Alloc &gt; Struct Template Reference

Inheritance diagram for std::\_\_future\_base::\_Result\_alloc< \_Res, \_Alloc >:



## Public Types

- using **\_\_allocator\_type** = \_\_alloc\_rebind< \_Alloc, [\\_Result\\_alloc](#) >
- typedef \_Res **result\_type**

## Public Member Functions

- **\_Result\_alloc** (const \_Alloc &\_\_a)
- void **\_M\_set** (const \_Res &\_\_res)
- void **\_M\_set** (\_Res &&\_\_res)
- \_Res & **\_M\_value** () noexcept

## Public Attributes

- [exception\\_ptr](#) **\_M\_error**

## 4.206.1 Detailed Description

```
template<typename _Res, typename _Alloc>
struct std::__future_base::__Result_alloc< _Res, _Alloc >
```

A result object that uses an allocator.

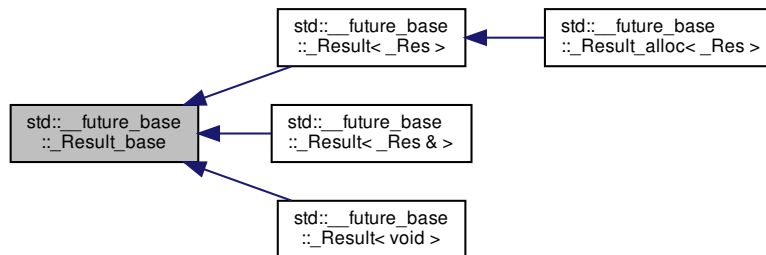
Definition at line 268 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

## 4.207 std::\_\_future\_base::\_\_Result\_base Struct Reference

Inheritance diagram for std::\_\_future\_base::\_\_Result\_base:



## Public Member Functions

- **\_Result\_base** (const [\\_Result\\_base](#) &)=delete
- virtual void **\_M\_destroy** ()=0
- [\\_Result\\_base](#) & **operator=** (const [\\_Result\\_base](#) &)=delete

## Public Attributes

- [exception\\_ptr](#) **\_M\_error**



#### 4.207.1 Detailed Description

Base class for results.

Definition at line 201 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

#### 4.208 `__gnu_debug::_Safe_container<_SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware>` Class Template Reference

Inherits `_SafeBase<_SafeContainer>`.

##### Public Member Functions

- `void _M_swap (_Safe_container &__x) noexcept`
- `_Safe_container & operator= (const _Safe_container &) noexcept`
- `_Safe_container & operator= (_Safe_container &&__x) noexcept`

##### Protected Member Functions

- `_Safe_container (const _Safe_container &)=default`
- `_Safe_container (_Safe_container &&)=default`
- `_Safe_container (_Safe_container &&__x, const _Alloc &__a)`
- `_Safe_container & _M_safe () noexcept`

#### 4.208.1 Detailed Description

```
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container<_SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware>
```

Safe class dealing with some allocator dependent operations.

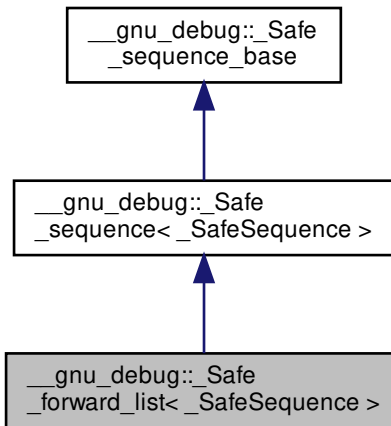
Definition at line 41 of file `safe_container.h`.

The documentation for this class was generated from the following file:

- [safe\\_container.h](#)

#### 4.209 `__gnu_debug::_Safe_forward_list<_SafeSequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_forward_list<_SafeSequence>`:



##### Public Member Functions

- `void _M_invalidate_if (_Predicate __pred)`
- `void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)`

##### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

##### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &) noexcept`

## 4.209.1 Detailed Description

```
template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list<_SafeSequence>
```

Special iterators swap and invalidation for forward\_list because of the before\_begin iterator.

Definition at line 56 of file debug/forward\_list.

## 4.209.2 Member Function Documentation

## 4.209.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

## 4.209.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

## 4.209.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence<map<\_Key, \_Tp, \_Compare, \_Allocator>>::\_M\_transfer\_from\_if().

#### 4.209.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.209.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 4.209.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.209.2.7 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_transfer_from_if (
 _Safe_sequence< _SafeSequence > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 4.209.3 Member Data Documentation

4.209.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

4.209.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

4.209.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

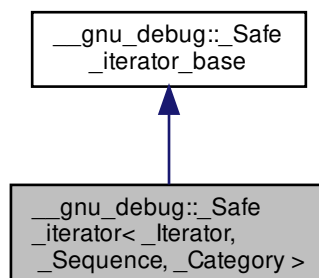
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

4.210 `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>`:



## Public Types

- typedef [\\_Safe\\_iterator](#)< [\\_Iterator](#), [\\_Sequence](#), [iterator\\_category](#) > [\\_Self](#)
- typedef [\\_Traits::difference\\_type](#) **difference\_type**
- typedef [\\_Traits::iterator\\_category](#) **iterator\_category**
- typedef [\\_Iterator](#) **iterator\_type**
- typedef [\\_Traits::pointer](#) **pointer**
- typedef [\\_Traits::reference](#) **reference**
- typedef [\\_Traits::value\\_type](#) **value\_type**

## Public Member Functions

- [\\_Safe\\_iterator](#) () noexcept
- [\\_Safe\\_iterator](#) ([\\_Iterator](#) \_\_i, const [\\_Safe\\_sequence\\_base](#) \* \_\_seq) noexcept
- [\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#) & \_\_x) noexcept
- [\\_Safe\\_iterator](#) ([\\_Safe\\_iterator](#) && \_\_x) noexcept
- template<typename [\\_MutableIterator](#) >  
[\\_Safe\\_iterator](#) (const [\\_Safe\\_iterator](#)< [\\_MutableIterator](#), [\\_Sequence](#), typename [\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_IsConstant::\\_\\_value](#) && [std::\\_\\_are\\_same](#)< [\\_MutableIterator](#), [\\_OtherIterator](#) >::\_\_value, [\\_Category](#) >::\_\_type > & \_\_x) noexcept
- void [\\_M\\_attach](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq)
- void [\\_M\\_attach\\_single](#) ([\\_Safe\\_sequence\\_base](#) \* \_\_seq)
- bool [\\_M\\_attached\\_to](#) (const [\\_Safe\\_sequence\\_base](#) \* \_\_seq) const
- bool [\\_M\\_before\\_dereferenceable](#) () const
- bool [\\_M\\_can\\_advance](#) ([difference\\_type](#) \_\_n, bool \_\_strict=false) const
- bool [\\_M\\_can\\_compare](#) (const [\\_Safe\\_iterator\\_base](#) & \_\_x) const throw ()
- bool [\\_M\\_dereferenceable](#) () const
- void [\\_M\\_detach\\_single](#) () throw ()
- [\\_Distance\\_traits](#)< [\\_Iterator](#) >::\_\_type [\\_M\\_get\\_distance\\_from\\_begin](#) () const
- [\\_Distance\\_traits](#)< [\\_Iterator](#) >::\_\_type [\\_M\\_get\\_distance\\_to](#) (const [\\_Safe\\_iterator](#) & \_\_rhs) const
- [\\_Distance\\_traits](#)< [\\_Iterator](#) >::\_\_type [\\_M\\_get\\_distance\\_to\\_end](#) () const
- [\\_\\_gnu\\_cxx::\\_\\_conditional\\_type](#)< [\\_IsConstant::\\_\\_value](#), const [\\_Sequence](#) \*, [\\_Sequence](#) \* >::\_\_type [\\_M\\_get\\_sequence](#) () const
- bool [\\_M\\_incrementable](#) () const
- void [\\_M\\_invalidate](#) ()
- bool [\\_M\\_is\\_before\\_begin](#) () const
- bool [\\_M\\_is\\_begin](#) () const
- bool [\\_M\\_is\\_beginnest](#) () const
- bool [\\_M\\_is\\_end](#) () const
- void [\\_M\\_reset](#) () throw ()
- bool [\\_M\\_singular](#) () const throw ()
- void [\\_M\\_unlink](#) () throw ()
- bool [\\_M\\_valid\\_range](#) (const [\\_Safe\\_iterator](#) & \_\_rhs, [std::pair](#)< [difference\\_type](#), [\\_Distance\\_precision](#) > & \_\_dist, bool \_\_check\_dereferenceable=true) const
- [\\_Iterator](#) & [base](#) () noexcept
- const [\\_Iterator](#) & [base](#) () const noexcept
- [operator \\_Iterator](#) () const noexcept
- [reference operator\\*](#) () const noexcept
- [\\_Safe\\_iterator](#) & [operator++](#) () noexcept
- [\\_Safe\\_iterator](#) [operator++](#) (int) noexcept
- [pointer operator->](#) () const noexcept
- [\\_Safe\\_iterator](#) & [operator=](#) (const [\\_Safe\\_iterator](#) & \_\_x) noexcept
- [\\_Safe\\_iterator](#) & [operator=](#) ([\\_Safe\\_iterator](#) && \_\_x) noexcept

## Static Public Member Functions

- static constexpr bool `_S_constant` ()

## Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

## Protected Types

- typedef std::\_\_are\_same< typename `_Sequence::Base::const_iterator`, `_Iterator` > **`_IsConstant`**
- typedef `__gnu_cxx::conditional_type< _IsConstant::__value, typename _Sequence::Base::iterator, typename _Sequence::Base::const_iterator >::__type` **`_OtherIterator`**

## Protected Member Functions

- **`_Safe_iterator`** (`_Iterator` \_\_i, `_Safe_sequence_base` \*\_\_seq, `_Attach_single`) noexcept
- void `_M_attach` (`_Safe_sequence_base` \*\_\_seq, bool \_\_constant)
- void `_M_attach_single` (`_Safe_sequence_base` \*\_\_seq, bool \_\_constant) throw ()
- void `_M_detach` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()

## Friends

- bool **`operator!=`** (const `_Self` &\_\_lhs, const `_Self` &\_\_rhs) noexcept
- template<typename `_IterR` >  
bool **`operator!=`** (const `_Self` &\_\_lhs, const `_Safe_iterator`< `_IterR`, `_Sequence`, `iterator_category` > &\_\_rhs) noexcept
- bool **`operator==`** (const `_Self` &\_\_lhs, const `_Self` &\_\_rhs) noexcept
- template<typename `_IterR` >  
bool **`operator==`** (const `_Self` &\_\_lhs, const `_Safe_iterator`< `_IterR`, `_Sequence`, `iterator_category` > &\_\_rhs) noexcept

## 4.210.1 Detailed Description

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
class __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>
```

Safe iterator wrapper.

The class template `_Safe_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Note that `_Iterator` must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before `_Iterator` get destroyed. Otherwise it would result in a data race.

Definition at line 61 of file `debug.h`.

#### 4.210.2 Constructor & Destructor Documentation

##### 4.210.2.1 `_Safe_iterator()` [1/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator () [inline],
[noexcept]
```

###### Postcondition

the iterator is singular and unattached

Definition at line 151 of file `safe_iterator.h`.

##### 4.210.2.2 `_Safe_iterator()` [2/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
 _Iterator __i,
 const _Safe_sequence_base * __seq) [inline], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

###### Precondition

`seq` is not NULL

###### Postcondition

this is not singular

Definition at line 160 of file `safe_iterator.h`.

##### 4.210.2.3 `_Safe_iterator()` [3/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __x) [inline], [noexcept]
```

Copy construction.

Definition at line 172 of file `safe_iterator.h`.



## 4.210.2.4 \_Safe\_iterator() [4/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
 _Safe_iterator< _Iterator, _Sequence, _Category > && __x) [inline], [noexcept]
```

Move construction.

## Postcondition

\_\_x is singular and unattached

Definition at line 190 of file safe\_iterator.h.

## 4.210.2.5 \_Safe\_iterator() [5/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
template<typename _MutableIterator >
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
 const _Safe_iterator< _MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<
_IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Category
>::__type > & __x) [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 210 of file safe\_iterator.h.

## 4.210.3 Member Function Documentation

## 4.210.3.1 \_M\_attach() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 _Safe_sequence_base * __seq,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_attach(), and \_↵\_\_gnu\_debug::\_Safe\_iterator\_base::\_Safe\_iterator\_base().

#### 4.210.3.2 `_M_attach()` [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach (
 _Safe_sequence_base * __seq) [inline]
```

Attach iterator to the given sequence.

Definition at line 374 of file `safe_iterator.h`.

#### 4.210.3.3 `_M_attach_single()` [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 _Safe_sequence_base * __seq,
 bool __constant) throw () [protected], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_attach_single()`.

#### 4.210.3.4 `_M_attach_single()` [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach_single (
 _Safe_sequence_base * __seq) [inline]
```

Likewise, but not thread-safe.

Definition at line 379 of file `safe_iterator.h`.

#### 4.210.3.5 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

#### 4.210.3.6 \_M\_before\_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_before_dereferenceable (
) const [inline]
```

Is the iterator before a dereferenceable one?

Definition at line 389 of file safe\_iterator.h.

#### 4.210.3.7 \_M\_can\_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const __Safe_iterator_base & __x) const throw () [inherited]
```

Can we compare this iterator to the given iterator \_\_x? Returns true if both iterators are nonsingular and reference the same sequence.

#### 4.210.3.8 \_M\_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Definition at line 384 of file safe\_iterator.h.

#### 4.210.3.9 \_M\_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.

#### 4.210.3.10 \_M\_detach\_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw () [inherited]
```

Likewise, but not thread-safe.

#### 4.210.3.11 \_M\_get\_mutex()

```
__gnu_cxx::mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_\_Safe\_iterator.

#### 4.210.3.12 `_M_incrementable()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_incrementable () const
[inline]
```

Is the iterator incrementable?

Definition at line 401 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_before_dereferenceable()`.

#### 4.210.3.13 `_M_invalidate()`

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

#### 4.210.3.14 `_M_is_before_begin()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_before_begin () const
[inline]
```

Is this iterator equal to the sequence's `before_begin()` iterator if any?

Definition at line 445 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_dereferenceable()`.

#### 4.210.3.15 `_M_is_begin()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_begin () const [inline]
```

Is this iterator equal to the sequence's `begin()` iterator?

Definition at line 434 of file `safe_iterator.h`.

#### 4.210.3.16 \_M\_is\_beginnest()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_beginnest () const
[inline]
```

Is this iterator equal to the sequence's before\_begin() iterator if any or begin() otherwise?

Definition at line 451 of file safe\_iterator.h.

#### 4.210.3.17 \_M\_is\_end()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_end () const [inline]
```

Is this iterator equal to the sequence's end() iterator?

Definition at line 439 of file safe\_iterator.h.

Referenced by \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_dereferenceable(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_incrementable().

#### 4.210.3.18 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset () throw () [inherited]
```

Reset all member variables

#### 4.210.3.19 \_M\_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw () [inherited]
```

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_incrementable().

#### 4.210.3.20 `_M_unlink()`

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () throw () [inline], [inherited]
```

Unlink itself

Definition at line 155 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_next`, and `__gnu_debug::_Safe_iterator_base::_M_prior`.

#### 4.210.3.21 `_S_constant()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
static constexpr bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_S_constant
() [inline], [static]
```

Determine if this is a constant iterator.

Definition at line 354 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_attach()`, and `↵  
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_attach_single()`.

#### 4.210.3.22 `base()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
_Iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::base () [inline],
[noexcept]
```

Return the underlying iterator.

Definition at line 361 of file `safe_iterator.h`.

Referenced by `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_before_↵  
dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_is_begin()`,  
and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_is_end()`.

#### 4.210.3.23 `operator _Iterator()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator _Iterator () const
[inline], [noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 370 of file `safe_iterator.h`.

#### 4.210.3.24 `operator*()`

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
reference __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator* () const
[inline], [noexcept]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 299 of file `safe_iterator.h`.

#### 4.210.3.25 `operator++()` [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
_Safe_iterator& __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator++ ()
[inline], [noexcept]
```

Iterator preincrement.

##### Precondition

iterator is incrementable

Definition at line 326 of file `safe_iterator.h`.

#### 4.210.3.26 `operator++()` [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
_Safe_iterator __gnu_debug::_Safe_iterator<_Iterator, _Sequence, _Category>::operator++ (↵
int) [inline], [noexcept]
```

Iterator postincrement.

##### Precondition

iterator is incrementable

Definition at line 341 of file `safe_iterator.h`.

#### 4.210.3.27 operator->()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator-> () const
[inline], [noexcept]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 312 of file safe\_iterator.h.

#### 4.210.3.28 operator=() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
 const _Safe_iterator< _Iterator, _Sequence, _Category > & __x) [inline], [noexcept]
```

Copy assignment.

Definition at line 232 of file safe\_iterator.h.

#### 4.210.3.29 operator=() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↵
traits<_Iterator>::iterator_category>
_Safe_iterator& __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
 _Safe_iterator< _Iterator, _Sequence, _Category > && __x) [inline], [noexcept]
```

Move assignment.

##### Postcondition

\_\_x is singular and unattached

Definition at line 264 of file safe\_iterator.h.

#### 4.210.4 Member Data Documentation



4.210.4.1 `_M_next`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

4.210.4.2 `_M_prior`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

4.210.4.3 `_M_sequence`

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be `NULL` to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

4.210.4.4 `_M_version`

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

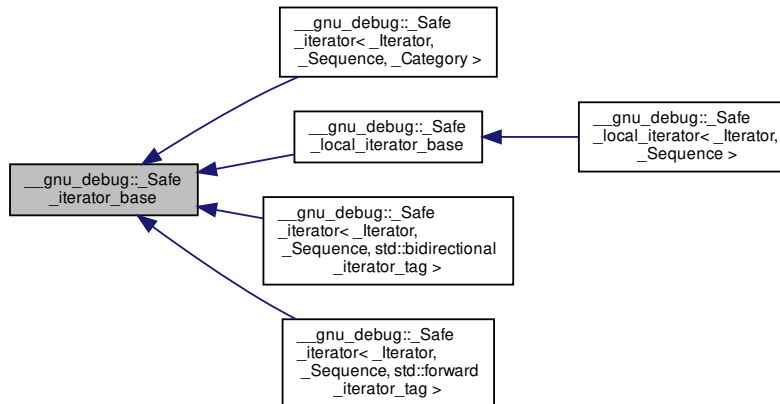
Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- `debug.h`
- `safe_iterator.h`
- `safe_iterator.tcc`

## 4.211 \_\_gnu\_debug::\_Safe\_iterator\_base Class Reference

Inheritance diagram for \_\_gnu\_debug::\_Safe\_iterator\_base:



### Public Member Functions

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_compare (const _Safe_iterator_base & __x) const throw ()`
- `void _M_detach_single () throw ()`
- `void _M_invalidate ()`
- `void _M_reset () throw ()`
- `bool _M_singular () const throw ()`
- `void _M_unlink () throw ()`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `_Safe_iterator_base ()`
- `_Safe_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_iterator_base (const _Safe_iterator_base & __x, bool __constant)`
- `void _M_attach (_Safe_sequence_base * __seq, bool __constant)`
- `void _M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`
- `void _M_detach ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

## Friends

- class **\_Safe\_sequence\_base**

### 4.211.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_iterator_base` base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_base.h`.

### 4.211.2 Constructor & Destructor Documentation

#### 4.211.2.1 `_Safe_iterator_base()` [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base () [inline], [protected]
```

Initializes the iterator and makes it singular.

Definition at line 78 of file `safe_base.h`.

#### 4.211.2.2 `_Safe_iterator_base()` [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
 const _Safe_sequence_base * __seq,
 bool __constant) [inline], [protected]
```

Initialize the iterator to reference the sequence pointed to by `__seq`. `__constant` is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 89 of file `safe_base.h`.

References `_M_attach()`.

#### 4.211.2.3 `_Safe_iterator_base()` [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
 const _Safe_iterator_base & __x,
 bool __constant) [inline], [protected]
```

Initializes the iterator to reference the same sequence that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 96 of file `safe_base.h`.

References `_M_attach()`, and `_M_sequence`.

### 4.211.3 Member Function Documentation

#### 4.211.3.1 `_M_attach()`

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
 _Safe_sequence_base * __seq,
 bool __constant) [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, std::forward_iterator_tag >::_M_attach()`, and `↔_Safe_iterator_base()`.

#### 4.211.3.2 `_M_attach_single()`

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
 _Safe_sequence_base * __seq,
 bool __constant) throw () [protected]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_iterator<_Iterator, _Sequence, std::forward_iterator_tag >::_M_attach_single()`.

#### 4.211.3.3 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `_M_sequence`.

#### 4.211.3.4 \_M\_can\_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const __Safe_iterator_base & __x) const throw ()
```

Can we compare this iterator to the given iterator \_\_x? Returns true if both iterators are nonsingular and reference the same sequence.

#### 4.211.3.5 \_M\_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.

#### 4.211.3.6 \_M\_detach\_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () throw ()
```

Likewise, but not thread-safe.

#### 4.211.3.7 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected]
```

For use in \_Safe\_iterator.

#### 4.211.3.8 \_M\_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file safe\_base.h.

References \_M\_version.

#### 4.211.3.9 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset () throw ()
```

Reset all member variables

#### 4.211.3.10 `_M_singular()`

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw ()
```

Is this iterator singular?

Referenced by `__gnu_debug::_check_singular_aux()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_dereferenceable()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`, and `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, std::forward_iterator_tag >::_M_incrementable()`.

#### 4.211.3.11 `_M_unlink()`

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () throw () [inline]
```

Unlink itself

Definition at line 155 of file `safe_base.h`.

References `_M_next`, and `_M_prior`.

### 4.211.4 Member Data Documentation

#### 4.211.4.1 `_M_next`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `_M_unlink()`.

#### 4.211.4.2 `_M_prior`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `_M_unlink()`.

4.211.4.3 `_M_sequence`

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence
```

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `_M_attached_to()`, `__gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `_Safe_iterator_base()`, and `__gnu_debug::Safe_local_iterator_base::_Safe_local_iterator_base()`.

4.211.4.4 `_M_version`

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

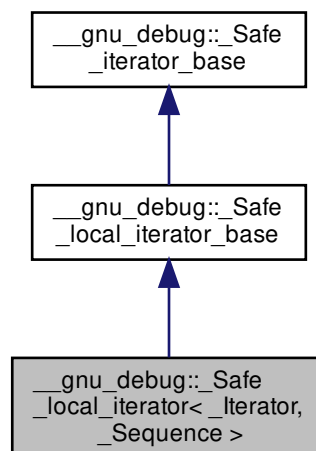
Referenced by `_M_invalidate()`, and `__gnu_debug::Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

4.212 `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence>`:



## Public Types

- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::iterator_category` **iterator\_category**
- typedef `_Iterator` **iterator\_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::value_type` **value\_type**

## Public Member Functions

- `_Safe_local_iterator` () noexcept
- `_Safe_local_iterator` (`_Iterator __i`, const `_Safe_sequence_base * __seq`)
- `_Safe_local_iterator` (const `_Safe_local_iterator` & `__x`) noexcept
- `_Safe_local_iterator` (`_Safe_local_iterator` && `__x`) noexcept
- template<typename `_MutableIterator` >  
`_Safe_local_iterator` (const `_Safe_local_iterator`< `_MutableIterator`, typename `__gnu_cxx::__enable_if< _IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Sequence >::__type >` & `__x`) noexcept
- void `_M_attach` (`_Safe_sequence_base * __seq`)
- void `_M_attach_single` (`_Safe_sequence_base * __seq`)
- bool `_M_attached_to` (const `_Safe_sequence_base * __seq`) const
- bool `_M_can_compare` (const `_Safe_iterator_base` & `__x`) const throw ()
- bool `_M_dereferenceable` () const
- `_Distance_traits< _Iterator >::__type` `_M_get_distance_to` (const `_Safe_local_iterator` & `__rhs`) const
- `__gnu_cxx::__conditional_type< _IsConstant::__value, const _Sequence *, _Sequence * >::__type` `_M_get_sequence` () const
- template<typename `_Other` >  
bool `_M_in_same_bucket` (const `_Safe_local_iterator`< `_Other`, `_Sequence >` & `__other`) const
- bool `_M_incrementable` () const
- void `_M_invalidate` ()
- bool `_M_is_begin` () const
- bool `_M_is_end` () const
- void `_M_reset` () throw ()
- bool `_M_singular` () const throw ()
- void `_M_unlink` () throw ()
- bool `_M_valid_range` (const `_Safe_local_iterator` & `__rhs`, `std::pair< difference_type, _Distance_precision >` & `__dist_info`) const
- `_Iterator` & `base` () noexcept
- const `_Iterator` & `base` () const noexcept
- size\_type `bucket` () const
- `operator _Iterator` () const
- reference `operator*` () const
- `_Safe_local_iterator` & `operator++` ()
- `_Safe_local_iterator` `operator++` (int)
- pointer `operator->` () const
- `_Safe_local_iterator` & `operator=` (const `_Safe_local_iterator` & `__x`)
- `_Safe_local_iterator` & `operator=` (`_Safe_local_iterator` && `__x`) noexcept



## Static Public Member Functions

- static constexpr bool `_S_constant` ()

## Public Attributes

- `_Safe_iterator_base` \* `_M_next`
- `_Safe_iterator_base` \* `_M_prior`
- `_Safe_sequence_base` \* `_M_sequence`
- unsigned int `_M_version`

## Protected Member Functions

- void `_M_attach` (`_Safe_sequence_base` \* `__seq`, bool `__constant`)
- void `_M_attach_single` (`_Safe_sequence_base` \* `__seq`, bool `__constant`) throw ()
- void `_M_detach` ()
- void `_M_detach_single` () throw ()
- `_Safe_unordered_container_base` \* `_M_get_container` () const noexcept
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()

## Friends

- bool **operator!=** (const `_Self` & `__lhs`, const `_OtherSelf` & `__rhs`) noexcept
- bool **operator!=** (const `_Self` & `__lhs`, const `_Self` & `__rhs`) noexcept
- bool **operator==** (const `_Self` & `__lhs`, const `_OtherSelf` & `__rhs`) noexcept
- bool **operator==** (const `_Self` & `__lhs`, const `_Self` & `__rhs`) noexcept

## 4.212.1 Detailed Description

```
template<typename _Iterator, typename _Sequence>
class __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>
```

Safe iterator wrapper.

The class template `_Safe_local_iterator` is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In addition to the basic iterator operations (which are validated, and then passed to the underlying iterator), `_Safe_local_iterator` has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.

Definition at line 83 of file `formatter.h`.

## 4.212.2 Constructor &amp; Destructor Documentation

**4.212.2.1** `_Safe_local_iterator()` [1/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator () [inline],
[noexcept]
```

**Postcondition**

the iterator is singular and unattached

Definition at line 102 of file `safe_local_iterator.h`.

**4.212.2.2** `_Safe_local_iterator()` [2/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
 _Iterator __i,
 const _Safe_sequence_base * __cont) [inline]
```

Safe iterator construction from an unsafe iterator and its sequence.

**Precondition**

`seq` is not NULL

**Postcondition**

this is not singular

Definition at line 111 of file `safe_local_iterator.h`.

**4.212.2.3** `_Safe_local_iterator()` [3/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
 const _Safe_local_iterator< _Iterator, _Sequence > & __x) [inline], [noexcept]
```

Copy construction.

Definition at line 122 of file `safe_local_iterator.h`.

4.212.2.4 `_Safe_local_iterator()` [4/5]

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
 _Safe_local_iterator< _Iterator, _Sequence > && __x) [inline], [noexcept]
```

Move construction.

**Postcondition**

`__x` is singular and unattached

Definition at line 139 of file `safe_local_iterator.h`.

4.212.2.5 `_Safe_local_iterator()` [5/5]

```
template<typename _Iterator , typename _Sequence >
template<typename _MutableIterator >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_Safe_local_iterator (
 const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _Is↵
Constant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Sequence >::__↵
__type > & __x) [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

Definition at line 158 of file `safe_local_iterator.h`.

**4.212.3 Member Function Documentation**4.212.3.1 `_M_attach()` [1/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
 _Safe_sequence_base * __seq,
 bool __constant) [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach()`, and `__gnu_debug::_Safe↵_local_iterator_base::_Safe_local_iterator_base()`.

#### 4.212.3.2 `_M_attach()` [2/2]

```
template<typename _Iterator , typename _Sequence >
void __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach (
 _Safe_sequence_base * __seq) [inline]
```

Attach iterator to the given sequence.

Definition at line 326 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator_base::_M_attach()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_S_constant()`.

#### 4.212.3.3 `_M_attach_single()` [1/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
 _Safe_sequence_base * __seq,
 bool __constant) throw () [protected], [inherited]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach_single()`.

#### 4.212.3.4 `_M_attach_single()` [2/2]

```
template<typename _Iterator , typename _Sequence >
void __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_attach_single (
 _Safe_sequence_base * __seq) [inline]
```

Likewise, but not thread-safe.

Definition at line 331 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator_base::_M_attach_single()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_S_constant()`.

#### 4.212.3.5 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

## 4.212.3.6 \_M\_can\_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const _Safe_iterator_base & __x) const throw () [inherited]
```

Can we compare this iterator to the given iterator \_\_x? Returns true if both iterators are nonsingular and reference the same sequence.

## 4.212.3.7 \_M\_dereferenceable()

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?

Definition at line 336 of file safe\_local\_iterator.h.

References \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_is\_end(), and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_singular().

## 4.212.3.8 \_M\_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.

## 4.212.3.9 \_M\_detach\_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single () throw () [protected], [inherited]
```

Likewise, but not thread-safe.

## 4.212.3.10 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_iterator.

## 4.212.3.11 \_M\_in\_same\_bucket()

```
template<typename _Iterator , typename _Sequence >
template<typename _Other >
bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_in_same_bucket (
 const _Safe_local_iterator<_Other, _Sequence> & __other) const [inline]
```

Is this iterator part of the same bucket as the other one?

Definition at line 371 of file safe\_local\_iterator.h.

References \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::bucket().

#### 4.212.3.12 `_M_incrementable()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable () const [inline]
```

Is the iterator incrementable?

Definition at line 341 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end()`, and `__gnu_debug::_Safe_iterator_base::_M_singular()`.

#### 4.212.3.13 `_M_invalidate()`

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_version`.

#### 4.212.3.14 `_M_is_begin()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_begin () const [inline]
```

Is this iterator equal to the sequence's `begin(bucket)` iterator?

Definition at line 361 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

#### 4.212.3.15 `_M_is_end()`

```
template<typename _Iterator , typename _Sequence >
bool __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_is_end () const [inline]
```

Is this iterator equal to the sequence's `end(bucket)` iterator?

Definition at line 365 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_dereferenceable()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::_M_incrementable()`.

## 4.212.3.16 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset () throw () [inherited]
```

Reset all member variables

## 4.212.3.17 \_M\_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw () [inherited]
```

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence, std::forward\_iterator\_tag>::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_incrementable(), and \_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence, std::forward\_iterator\_tag>::\_M\_incrementable().

## 4.212.3.18 \_M\_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () throw () [inline], [inherited]
```

Unlink itself

Definition at line 155 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior.

## 4.212.3.19 \_S\_constant()

```
template<typename _Iterator , typename _Sequence >
static constexpr bool __gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_S_constant ()
[inline], [static]
```

Determine if this is a constant iterator.

Definition at line 300 of file safe\_local\_iterator.h.

Referenced by \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_attach(), and \_\_gnu\_debug::\_Safe\_local\_iterator<\_Iterator, \_Sequence>::\_M\_attach\_single().

#### 4.212.3.20 base()

```
template<typename _Iterator , typename _Sequence >
_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base () [inline], [noexcept]
```

Return the underlying iterator.

Definition at line 307 of file `safe_local_iterator.h`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_begin()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_end()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket()`.

#### 4.212.3.21 bucket()

```
template<typename _Iterator , typename _Sequence >
size_type __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::bucket () const [inline]
```

Return the bucket.

Definition at line 316 of file `safe_local_iterator.h`.

References `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::base()`.

Referenced by `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_in_same_bucket()`, `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_begin()`, and `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::M_is_end()`.

#### 4.212.3.22 operator\_iterator()

```
template<typename _Iterator , typename _Sequence >
__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator_iterator () const [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

Definition at line 322 of file `safe_local_iterator.h`.

#### 4.212.3.23 operator\*()

```
template<typename _Iterator , typename _Sequence >
reference __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator* () const [inline]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 244 of file `safe_local_iterator.h`.



#### 4.212.3.24 `operator++()` [1/2]

```
template<typename _Iterator , typename _Sequence >
__Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ ()
[inline]
```

Iterator preincrement.

##### Precondition

iterator is incrementable

Definition at line 271 of file `safe_local_iterator.h`.

#### 4.212.3.25 `operator++()` [2/2]

```
template<typename _Iterator , typename _Sequence >
__Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator++ (
 int) [inline]
```

Iterator postincrement.

##### Precondition

iterator is incrementable

Definition at line 286 of file `safe_local_iterator.h`.

#### 4.212.3.26 `operator->()`

```
template<typename _Iterator , typename _Sequence >
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator-> () const [inline]
```

Iterator dereference.

##### Precondition

iterator is dereferenceable

Definition at line 257 of file `safe_local_iterator.h`.

**4.212.3.27 operator=()** [1/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator= (
 const _Safe_local_iterator< _Iterator, _Sequence > & __x) [inline]
```

Copy assignment.

Definition at line 179 of file `safe_local_iterator.h`.

**4.212.3.28 operator=()** [2/2]

```
template<typename _Iterator , typename _Sequence >
_Safe_local_iterator& __gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >::operator= (
 _Safe_local_iterator< _Iterator, _Sequence > && __x) [inline], [noexcept]
```

Move assignment.

**Postcondition**

\_\_x is singular and unattached

Definition at line 210 of file `safe_local_iterator.h`.

**4.212.4 Member Data Documentation****4.212.4.1 \_M\_next**

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

**4.212.4.2 \_M\_prior**

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence != NULL`.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

## 4.212.4.3 \_M\_sequence

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base()`.

## 4.212.4.4 \_M\_version

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

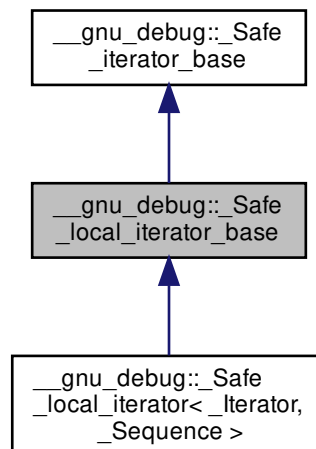
Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_local\\_iterator.h](#)
- [safe\\_local\\_iterator.tcc](#)

## 4.213 \_\_gnu\_debug::\_Safe\_local\_iterator\_base Class Reference

Inheritance diagram for `__gnu_debug::_Safe_local_iterator_base`:



### Public Member Functions

- `bool _M_attached_to (const _Safe_sequence_base * __seq) const`
- `bool _M_can_compare (const _Safe_iterator_base & __x) const throw ()`
- `void _M_invalidate ()`
- `void _M_reset () throw ()`
- `bool _M_singular () const throw ()`
- `void _M_unlink () throw ()`

### Public Attributes

- `_Safe_iterator_base * _M_next`
- `_Safe_iterator_base * _M_prior`
- `_Safe_sequence_base * _M_sequence`
- `unsigned int _M_version`

### Protected Member Functions

- `_Safe_local_iterator_base ()`
- `_Safe_local_iterator_base (const _Safe_sequence_base * __seq, bool __constant)`
- `_Safe_local_iterator_base (const _Safe_local_iterator_base & __x, bool __constant)`
- `void _M_attach (_Safe_sequence_base * __seq, bool __constant)`
- `void _M_attach_single (_Safe_sequence_base * __seq, bool __constant) throw ()`
- `void _M_detach ()`
- `void _M_detach_single () throw ()`
- `_Safe_unordered_container_base * _M_get_container () const noexcept`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`

#### 4.213.1 Detailed Description

Basic functionality for a *safe* iterator.

The `_Safe_local_iterator_base` base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.

This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

Definition at line 50 of file `safe_unordered_base.h`.

#### 4.213.2 Constructor & Destructor Documentation

## 4.213.2.1 \_Safe\_local\_iterator\_base() [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base () [inline], [protected]
```

Initializes the iterator and makes it singular.

Definition at line 54 of file safe\_unordered\_base.h.

## 4.213.2.2 \_Safe\_local\_iterator\_base() [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
 const _Safe_sequence_base * __seq,
 bool __constant) [inline], [protected]
```

Initialize the iterator to reference the container pointed to by `__seq`. `__constant` is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that `__seq` may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference `__seq` and be nonsingular.

Definition at line 64 of file safe\_unordered\_base.h.

References `_M_attach()`.

## 4.213.2.3 \_Safe\_local\_iterator\_base() [3/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
 const _Safe_local_iterator_base & __x,
 bool __constant) [inline], [protected]
```

Initializes the iterator to reference the same container that `__x` does. `__constant` is true if this is a constant iterator, and false if it is mutable.

Definition at line 70 of file safe\_unordered\_base.h.

References `_M_attach()`, and `__gnu_debug::_Safe_iterator_base::_M_sequence`.

## 4.213.3 Member Function Documentation

## 4.213.3.1 \_M\_attach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
 _Safe_sequence_base * __seq,
 bool __constant) [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach()`, and `_Safe_local_iterator_↔base()`.

#### 4.213.3.2 `_M_attach_single()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
 _Safe_sequence_base * __seq,
 bool __constant) throw () [protected]
```

Likewise, but not thread-safe.

Referenced by `__gnu_debug::_Safe_local_iterator<_Iterator, _Sequence>::_M_attach_single()`.

#### 4.213.3.3 `_M_attached_to()`

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
 const _Safe_sequence_base * __seq) const [inline], [inherited]
```

Determines if we are attached to the given sequence.

Definition at line 131 of file `safe_base.h`.

References `__gnu_debug::_Safe_iterator_base::_M_sequence`.

#### 4.213.3.4 `_M_can_compare()`

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
 const _Safe_iterator_base & __x) const throw () [inherited]
```

Can we compare this iterator to the given iterator `__x`? Returns true if both iterators are nonsingular and reference the same sequence.

#### 4.213.3.5 `_M_detach()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach () [protected]
```

Detach the iterator for whatever container it is attached to, if any.

#### 4.213.3.6 `_M_detach_single()`

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single () throw () [protected]
```

Likewise, but not thread-safe.

#### 4.213.3.7 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_iterator_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_iterator`.

#### 4.213.3.8 \_M\_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.

Definition at line 146 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_version.

#### 4.213.3.9 \_M\_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset () throw () [inherited]
```

Reset all member variables

#### 4.213.3.10 \_M\_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const throw () [inherited]
```

Is this iterator singular?

Referenced by \_\_gnu\_debug::\_check\_singular\_aux(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_dereferenceable(), \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >::\_M\_incrementable(), and \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, std::forward\_iterator\_tag >::\_M\_incrementable().

#### 4.213.3.11 \_M\_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () throw () [inline], [inherited]
```

Unlink itself

Definition at line 155 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_next, and \_\_gnu\_debug::\_Safe\_iterator\_base::\_M\_prior.

### 4.213.4 Member Data Documentation

#### 4.213.4.1 `_M_next`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Definition at line 74 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

#### 4.213.4.2 `_M_prior`

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when `_M_sequence` != NULL.

Definition at line 70 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, and `__gnu_debug::_Safe_iterator_base::_M_unlink()`.

#### 4.213.4.3 `_M_sequence`

```
_Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator.

Definition at line 57 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_attached_to()`, `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`, `__gnu_debug::_Safe_iterator_base::_Safe_iterator_base()`, and `_Safe_local_iterator_base()`.

#### 4.213.4.4 `_M_version`

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by `_M_sequence` for the iterator to be non-singular.

Definition at line 66 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_iterator_base::_M_invalidate()`, and `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

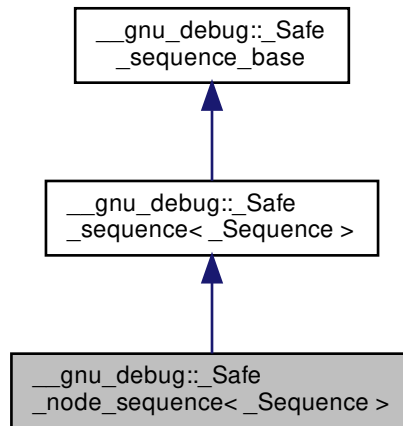
The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)



4.214 `__gnu_debug::_Safe_node_sequence<_Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_node_sequence<_Sequence>`:



## Public Member Functions

- `template<typename _Predicate>`  
`void _M_invalidate_if (_Predicate __pred)`
- `template<typename _Predicate>`  
`void _M_transfer_from_if (_Safe_sequence &__from, _Predicate __pred)`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular ()`
- `void _M_swap (_Safe_sequence_base &__x) noexcept`

#### 4.214.1 Detailed Description

```
template<typename _Sequence>
class __gnu_debug::_Safe_node_sequence< _Sequence >
```

Like `_Safe_sequence` but with a special `_M_invalidate_all` implementation not invalidating past-the-end iterators. Used by node based sequence.

Definition at line 131 of file `safe_sequence.h`.

#### 4.214.2 Member Function Documentation

##### 4.214.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base()`.

##### 4.214.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

##### 4.214.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

## 4.214.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 4.214.2.5 \_M\_invalidate\_if()

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

## 4.214.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.214.2.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.214.2.8 \_M\_transfer\_from\_if()

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
 _Safe_sequence< _Sequence > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

#### 4.214.3 Member Data Documentation

##### 4.214.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.214.3.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.214.3.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

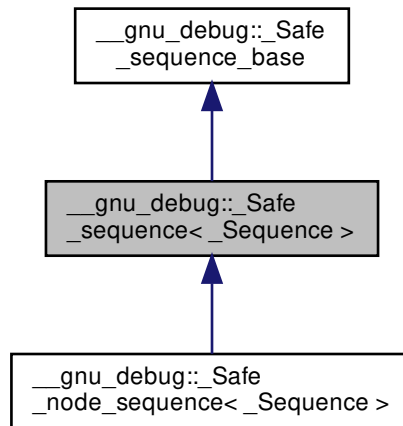
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [safe\\_sequence.h](#)

4.215 `__gnu_debug::_Safe_sequence<_Sequence>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_sequence<_Sequence>`:



## Public Member Functions

- `template<typename _Predicate>`  
`void \_M\_invalidate\_if (_Predicate __pred)`
- `template<typename _Predicate>`  
`void \_M\_transfer\_from\_if (\_Safe\_sequence &__from, _Predicate __pred)`

## Public Attributes

- `\_Safe\_iterator\_base * \_M\_const\_iterators`
- `\_Safe\_iterator\_base * \_M\_iterators`
- `unsigned int \_M\_version`

## Protected Member Functions

- `void \_M\_detach\_all ()`
- `void \_M\_detach\_singular ()`
- `\_\_gnu\_cxx::\_\_mutex & \_M\_get\_mutex () throw ()`
- `void \_M\_invalidate\_all () const`
- `void \_M\_revalidate\_singular ()`
- `void \_M\_swap (\_Safe\_sequence\_base &__x) noexcept`

#### 4.215.1 Detailed Description

```
template<typename _Sequence>
class __gnu_debug::_Safe_sequence< _Sequence >
```

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template `_Safe_sequence` simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this sequence. Iterators will then be tracked automatically.

Definition at line 86 of file `formatter.h`.

#### 4.215.2 Member Function Documentation

##### 4.215.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

##### 4.215.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

##### 4.215.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

4.215.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

4.215.2.5 `_M_invalidate_if()`

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence<_Sequence>::_M_invalidate_if (
 _Predicate __pred)
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

4.215.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.215.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

4.215.2.8 `_M_transfer_from_if()`

```
template<typename _Sequence >
template<typename _Predicate >
void __gnu_debug::_Safe_sequence<_Sequence>::_M_transfer_from_if (
 _Safe_sequence<_Sequence> & __from,
 _Predicate __pred)
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

#### 4.215.3 Member Data Documentation

##### 4.215.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.215.3.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.215.3.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

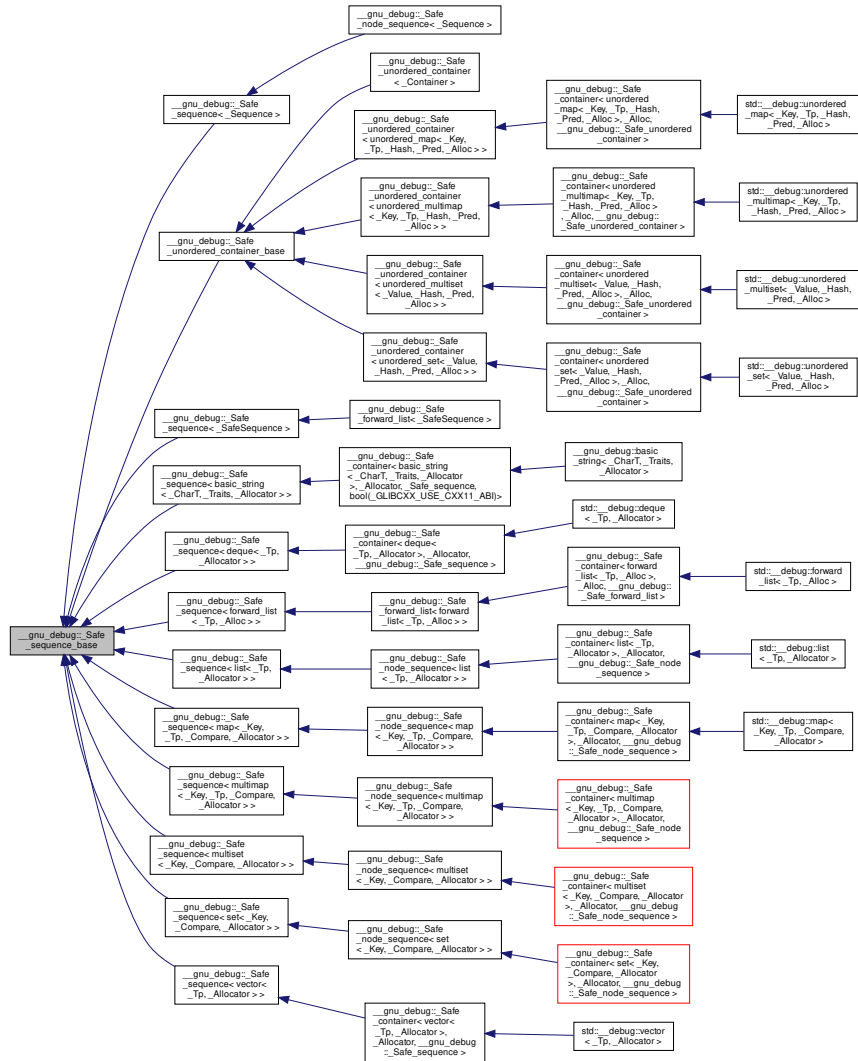
The documentation for this class was generated from the following files:

- [formatter.h](#)
- [safe\\_sequence.h](#)
- [safe\\_sequence.tcc](#)



## 4.216 \_\_gnu\_debug:: Safe\_sequence\_base Class Reference

Inheritance diagram for \_\_gnu\_debug:: Safe\_sequence\_base:



## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- [\\_Safe\\_sequence\\_base](#) (const [\\_Safe\\_sequence\\_base](#) &) noexcept

- `_Safe_sequence_base` (`_Safe_sequence_base` &&\_\_seq) noexcept
- `~_Safe_sequence_base` ()
- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_invalidate_all` () const
- `void _M_revalidate_singular` ()
- `void _M_swap` (`_Safe_sequence_base` &\_\_x) noexcept

#### Friends

- class `_Safe_iterator_base`

#### 4.216.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.

The `_Safe_sequence_base` class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from `_Safe_sequence_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 188 of file `safe_base.h`.

#### 4.216.2 Constructor & Destructor Documentation

##### 4.216.2.1 `~_Safe_sequence_base()`

```
__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base () [inline], [protected]
```

Notify all iterators that reference this sequence that the sequence is being destroyed.

Definition at line 220 of file `safe_base.h`.

References `_M_detach_all()`.

#### 4.216.3 Member Function Documentation

#### 4.216.3.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected]
```

Detach all iterators, leaving them singular.

Referenced by `~_Safe_sequence_base()`.

#### 4.216.3.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 4.216.3.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.216.3.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `_M_version`.

#### 4.216.3.5 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.216.3.6 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.216.4 Member Data Documentation

#### 4.216.4.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.216.4.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.216.4.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

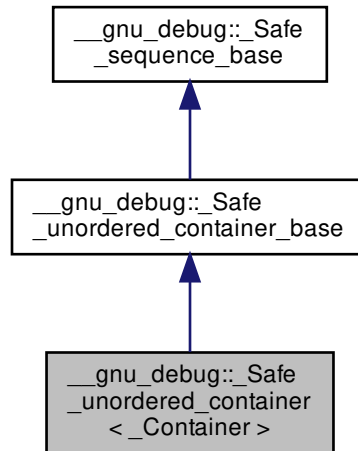
Referenced by `_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [safe\\_base.h](#)

4.217 `__gnu_debug::_Safe_unordered_container<_Container>` Class Template Reference

Inheritance diagram for `__gnu_debug::_Safe_unordered_container<_Container>`:



## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_local\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_local\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- template<typename [\\_Predicate](#)>  
void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- template<typename [\\_Predicate](#)>  
void [\\_M\\_invalidate\\_local\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_invalidate\\_locals](#) ()
- void [\\_M\\_revalidate\\_singular](#) ()
- void [\\_M\\_swap](#) ([\\_Safe\\_unordered\\_container\\_base](#) &\_\_x) noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

#### 4.217.1 Detailed Description

```
template<typename _Container>
class __gnu_debug::_Safe_unordered_container< _Container >
```

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.

The class template `_Safe_unordered_container` simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template `_Safe_iterator` for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template `_Safe_local_iterator` for this container. Iterators will then be tracked automatically.

Definition at line 58 of file `safe_unordered_container.h`.

#### 4.217.2 Member Function Documentation

##### 4.217.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

##### 4.217.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

##### 4.217.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

4.217.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

4.217.2.5 `_M_invalidate_if()`

```
template<typename _Container >
template<typename _Predicate >
void __gnu_debug::_Safe_unordered_container<_Container>::_M_invalidate_if (
 _Predicate __pred) [protected]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

4.217.2.6 `_M_invalidate_local_if()`

```
template<typename _Container >
template<typename _Predicate >
void __gnu_debug::_Safe_unordered_container<_Container>::_M_invalidate_local_if (
 _Predicate __pred) [protected]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

4.217.2.7 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

4.217.2.8 `_M_swap()` [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.217.2.9 `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.217.3 Member Data Documentation

#### 4.217.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.217.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 4.217.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.



#### 4.217.3.4 \_M\_local\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file safe\_unordered\_base.h.

#### 4.217.3.5 \_M\_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file safe\_base.h.

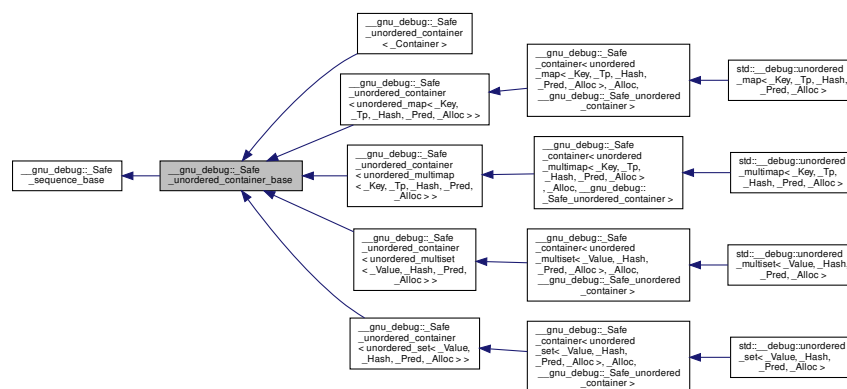
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following files:

- `safe_unordered_container.h`
- `safe_unordered_container.tcc`

## 4.218 `__gnu_debug::Safe_unordered_container_base` Class Reference

Inheritance diagram for `gnu_debug:: Safe_unordered_container_base`:



## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_const_local_iterators`
- `_Safe_iterator_base * _M_iterators`
- `_Safe_iterator_base * _M_local_iterators`
- `unsigned int M version`

### Protected Member Functions

- `_Safe_unordered_container_base` (const `_Safe_unordered_container_base` &) noexcept
- `_Safe_unordered_container_base` (`_Safe_unordered_container_base` &&\_\_x) noexcept
- `~_Safe_unordered_container_base` () noexcept
- `void _M_detach_all` ()
- `void _M_detach_singular` ()
- `__gnu_cxx::__mutex & _M_get_mutex` () throw ()
- `void _M_invalidate_all` () const
- `void _M_revalidate_singular` ()
- `void _M_swap` (`_Safe_unordered_container_base` &\_\_x) noexcept
- `void _M_swap` (`_Safe_sequence_base` &\_\_x) noexcept

### Friends

- class `_Safe_local_iterator_base`

#### 4.218.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.

The `_Safe_unordered_container_base` class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from `_Safe_unordered_container_base` publicly, so that safe iterators (which inherit `_Safe_iterator_base`) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.

This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

Definition at line 120 of file `safe_unordered_base.h`.

#### 4.218.2 Constructor & Destructor Documentation

##### 4.218.2.1 `~_Safe_unordered_container_base()`

```
__gnu_debug::_Safe_unordered_container_base::~~_Safe_unordered_container_base () [inline], [protected],
[noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.

Definition at line 151 of file `safe_unordered_base.h`.

#### 4.218.3 Member Function Documentation

#### 4.218.3.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected]
```

Detach all iterators, leaving them singular.

#### 4.218.3.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

#### 4.218.3.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.218.3.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

#### 4.218.3.5 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.218.3.6 `_M_swap()` [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.218.3.7 `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.218.4 Member Data Documentation

#### 4.218.4.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.218.4.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 4.218.4.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

4.218.4.4 `_M_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

4.218.4.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [safe\\_unordered\\_base.h](#)

4.219 `__gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence>` Class Template Reference

## Protected Member Functions

- `_Safe_vector` (const [\\_Safe\\_vector](#) &) noexcept
- `_Safe_vector` (size\_type \_\_n) noexcept
- `_Safe_vector` ([\\_Safe\\_vector](#) &&\_\_x) noexcept
- `bool _M_requires_reallocation` (size\_type \_\_elements) const noexcept
- `void _M_update_guaranteed_capacity` () noexcept
- `_Safe_vector & operator=` (const [\\_Safe\\_vector](#) &) noexcept
- `_Safe_vector & operator=` ([\\_Safe\\_vector](#) &&\_\_x) noexcept

## Protected Attributes

- size\_type `_M_guaranteed_capacity`

## 4.219.1 Detailed Description

```
template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug::_Safe_vector<_SafeSequence, _BaseSequence>
```

Base class for Debug Mode vector.

Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.

Definition at line 55 of file `debug/vector`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

#### 4.220 `__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >` Struct Template Reference

##### Public Member Functions

- void **operator()** ( \_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

##### 4.220.1 Detailed Description

```
template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
```

Stable sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1007 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.221 `__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >` Struct Template Reference

##### Public Member Functions

- void **operator()** ( \_RAIter \_\_first, \_RAIter \_\_last, \_StrictWeakOrdering \_\_comp)

##### 4.221.1 Detailed Description

```
template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >
```

Non-`__stable` sorting functor.

Used to reduce code instantiation in `multiway_merge_sampling_splitting`.

Definition at line 1020 of file `multiway_merge.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_merge.h](#)

#### 4.222 `std::__detail::_Scanner< _CharT >` Class Template Reference

Inherits `std::__detail::_ScannerBase`.

## Public Types

- typedef const [std::ctype](#)<\_CharT> **\_CtypeT**
- typedef [regex\\_constants::syntax\\_option\\_type](#) **\_FlagT**
- typedef const \_CharT \* **\_IterT**
- typedef [std::basic\\_string](#)<\_CharT> **\_StringT**
- enum **\_TokenT** : unsigned {  
     **\_S\_token\_anychar**, **\_S\_token\_ord\_char**, **\_S\_token\_oct\_num**, **\_S\_token\_hex\_num**,  
     **\_S\_token\_backref**, **\_S\_token\_subexpr\_begin**, **\_S\_token\_subexpr\_no\_group\_begin**, **\_S\_token\_subexpr\_**  
     **\_lookahead\_begin**,  
     **\_S\_token\_subexpr\_end**, **\_S\_token\_bracket\_begin**, **\_S\_token\_bracket\_neg\_begin**, **\_S\_token\_bracket\_**  
     **end**,  
     **\_S\_token\_interval\_begin**, **\_S\_token\_interval\_end**, **\_S\_token\_quoted\_class**, **\_S\_token\_char\_class\_name**,  
     **\_S\_token\_collsymbol**, **\_S\_token\_equiv\_class\_name**, **\_S\_token\_opt**, **\_S\_token\_or**,  
     **\_S\_token\_closure0**, **\_S\_token\_closure1**, **\_S\_token\_line\_begin**, **\_S\_token\_line\_end**,  
     **\_S\_token\_word\_bound**, **\_S\_token\_comma**, **\_S\_token\_dup\_count**, **\_S\_token\_eof**,  
     **\_S\_token\_bracket\_dash**, **\_S\_token\_unknown** }

## Public Member Functions

- **\_Scanner** (**\_IterT** \_\_begin, **\_IterT** \_\_end, **\_FlagT** \_\_flags, [std::locale](#) \_\_loc)
- void **\_M\_advance** ()
- **\_TokenT** **\_M\_get\_token** () const
- const [\\_StringT](#) & **\_M\_get\_value** () const

## Protected Types

- enum **\_StateT** { **\_S\_state\_normal**, **\_S\_state\_in\_brace**, **\_S\_state\_in\_bracket** }

## Protected Member Functions

- const char \* **\_M\_find\_escape** (char \_\_c)
- bool **\_M\_is\_awk** () const
- bool **\_M\_is\_basic** () const
- bool **\_M\_is\_ecma** () const
- bool **\_M\_is\_extended** () const
- bool **\_M\_is\_grep** () const

## Protected Attributes

- bool **\_M\_at\_bracket\_start**
- const [std::pair](#)< char, char > **\_M\_awk\_escape\_tbl** [11]
- const char \* **\_M\_basic\_spec\_char**
- const [std::pair](#)< char, char > **\_M\_ecma\_escape\_tbl** [8]
- const char \* **\_M\_ecma\_spec\_char**
- const [std::pair](#)< char, char > \* **\_M\_escape\_tbl**
- const char \* **\_M\_extended\_spec\_char**
- **\_FlagT** **\_M\_flags**
- const char \* **\_M\_spec\_char**
- **\_StateT** **\_M\_state**
- **\_TokenT** **\_M\_token**
- const [std::pair](#)< char, **\_TokenT** > **\_M\_token\_tbl** [9]

#### 4.222.1 Detailed Description

```
template<typename _CharT>
class std::__detail::_Scanner< _CharT >
```

Scans an input range for regex tokens.

The `_Scanner` class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

Definition at line 210 of file `regex_scanner.h`.

#### 4.222.2 Member Enumeration Documentation

##### 4.222.2.1 `_TokenT`

```
enum std::__detail::_ScannerBase::_TokenT : unsigned [inherited]
```

Token types returned from the scanner.

Definition at line 46 of file `regex_scanner.h`.

The documentation for this class was generated from the following files:

- [regex\\_scanner.h](#)
- [regex\\_scanner.tcc](#)

#### 4.223 `__gnu_debug::_Sequence_traits< _Sequence >` Struct Template Reference

##### Public Types

- `typedef _Distance_traits< typename _Sequence::iterator > _DistTraits`

##### Static Public Member Functions

- `static \_DistTraits::\_type _S_size (const _Sequence &__seq)`



## 4.223.1 Detailed Description

```
template<typename _Sequence>
struct __gnu_debug::_Sequence_traits< _Sequence >
```

Sequence traits giving the size of a container if possible.

Definition at line 85 of file `safe_iterator.h`.

The documentation for this struct was generated from the following file:

- [safe\\_iterator.h](#)

## 4.224 \_\_gnu\_parallel::\_Settings Struct Reference

## Static Public Member Functions

- static const [\\_Settings](#) & [get](#) () throw ()
- static void [set](#) ([\\_Settings](#) &) throw ()

## Public Attributes

- [\\_SequenceIndex](#) [accumulate\\_minimal\\_n](#)
- unsigned int [adjacent\\_difference\\_minimal\\_n](#)
- [\\_AlgorithmStrategy](#) [algorithm\\_strategy](#)
- unsigned int [cache\\_line\\_size](#)
- [\\_SequenceIndex](#) [count\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [fill\\_minimal\\_n](#)
- [\\_FindAlgorithm](#) [find\\_algorithm](#)
- double [find\\_increasing\\_factor](#)
- [\\_SequenceIndex](#) [find\\_initial\\_block\\_size](#)
- [\\_SequenceIndex](#) [find\\_maximum\\_block\\_size](#)
- float [find\\_scale\\_factor](#)
- [\\_SequenceIndex](#) [find\\_sequential\\_search\\_size](#)
- [\\_SequenceIndex](#) [for\\_each\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [generate\\_minimal\\_n](#)
- unsigned long long [L1\\_cache\\_size](#)
- unsigned long long [L2\\_cache\\_size](#)
- [\\_SequenceIndex](#) [max\\_element\\_minimal\\_n](#)
- [\\_SequenceIndex](#) [merge\\_minimal\\_n](#)
- unsigned int [merge\\_oversampling](#)
- [\\_SplittingAlgorithm](#) [merge\\_splitting](#)
- [\\_SequenceIndex](#) [min\\_element\\_minimal\\_n](#)
- [\\_MultiwayMergeAlgorithm](#) [multiway\\_merge\\_algorithm](#)
- int [multiway\\_merge\\_minimal\\_k](#)
- [\\_SequenceIndex](#) [multiway\\_merge\\_minimal\\_n](#)
- unsigned int [multiway\\_merge\\_oversampling](#)
- [\\_SplittingAlgorithm](#) [multiway\\_merge\\_splitting](#)

- [\\_SequenceIndex nth\\_element\\_minimal\\_n](#)
- [\\_SequenceIndex partial\\_sort\\_minimal\\_n](#)
- [\\_PartialSumAlgorithm](#) **partial\_sum\_algorithm**
- float [partial\\_sum\\_dilation](#)
- unsigned int [partial\\_sum\\_minimal\\_n](#)
- double [partition\\_chunk\\_share](#)
- [\\_SequenceIndex partition\\_chunk\\_size](#)
- [\\_SequenceIndex partition\\_minimal\\_n](#)
- [\\_SequenceIndex qsb\\_steals](#)
- unsigned int [random\\_shuffle\\_minimal\\_n](#)
- [\\_SequenceIndex replace\\_minimal\\_n](#)
- [\\_SequenceIndex search\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_difference\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_intersection\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_symmetric\\_difference\\_minimal\\_n](#)
- [\\_SequenceIndex set\\_union\\_minimal\\_n](#)
- [\\_SortAlgorithm](#) **sort\_algorithm**
- [\\_SequenceIndex sort\\_minimal\\_n](#)
- unsigned int [sort\\_mwms\\_oversampling](#)
- unsigned int [sort\\_qs\\_num\\_samples\\_preset](#)
- [\\_SequenceIndex sort\\_qsb\\_base\\_case\\_maximal\\_n](#)
- [\\_SplittingAlgorithm](#) **sort\_splitting**
- unsigned int [TLB\\_size](#)
- [\\_SequenceIndex transform\\_minimal\\_n](#)
- [\\_SequenceIndex unique\\_copy\\_minimal\\_n](#)
- [\\_SequenceIndex](#) **workstealing\_chunk\_size**

#### 4.224.1 Detailed Description

class `_Settings` Run-time settings for the parallel mode including all tunable parameters.

Definition at line 122 of file `settings.h`.

#### 4.224.2 Member Function Documentation

##### 4.224.2.1 `get()`

```
static const _Settings& __gnu_parallel::_Settings::get () throw () [static]
```

Get the global settings.

Referenced by `__gnu_parallel::_find_template()`, `__gnu_parallel::_for_each_template_random_access_workstealing()`, `__gnu_parallel::_parallel_nth_element()`, `__gnu_parallel::_parallel_partial_sum()`, `__gnu_parallel::_parallel_↵  
partial_sum_linear()`, `__gnu_parallel::_parallel_partition()`, `__gnu_parallel::_parallel_random_shuffle_drs()`, `__gnu_↵  
_parallel::_parallel_sort()`, `__gnu_parallel::_qsb_local_sort_with_helping()`, `__gnu_parallel::_sequential_random_↵  
_shuffle()`, `__gnu_parallel::_multiway_merge_sampling_splitting()`, `__gnu_parallel::_parallel_sort_mwms()`, and `__gnu_↵  
_parallel::_parallel_sort_mwms_pu()`.

#### 4.224.2.2 set()

```
static void __gnu_parallel::_Settings::set (
 _Settings &) throw () [static]
```

Set the global settings.

#### 4.224.3 Member Data Documentation

##### 4.224.3.1 accumulate\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::accumulate\_minimal\_n

Minimal input size for accumulate.

Definition at line 138 of file settings.h.

##### 4.224.3.2 adjacent\_difference\_minimal\_n

unsigned int \_\_gnu\_parallel::\_Settings::adjacent\_difference\_minimal\_n

Minimal input size for adjacent\_difference.

Definition at line 141 of file settings.h.

##### 4.224.3.3 cache\_line\_size

unsigned int \_\_gnu\_parallel::\_Settings::cache\_line\_size

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.

Definition at line 264 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_workstealing().

##### 4.224.3.4 count\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::count\_minimal\_n

Minimal input size for count and count\_if.

Definition at line 144 of file settings.h.

#### 4.224.3.5 fill\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::fill\_minimal\_n

Minimal input size for fill.

Definition at line 147 of file settings.h.

#### 4.224.3.6 find\_increasing\_factor

double \_\_gnu\_parallel::\_Settings::find\_increasing\_factor

Block size increase factor for find.

Definition at line 150 of file settings.h.

#### 4.224.3.7 find\_initial\_block\_size

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::find\_initial\_block\_size

Initial block size for find.

Definition at line 153 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.224.3.8 find\_maximum\_block\_size

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::find\_maximum\_block\_size

Maximal block size for find.

Definition at line 156 of file settings.h.

#### 4.224.3.9 find\_scale\_factor

float \_\_gnu\_parallel::\_Settings::find\_scale\_factor

Block size scale-down factor with respect to current position.

Definition at line 275 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.224.3.10 find\_sequential\_search\_size

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::find\_sequential\_search\_size

Start with looking for this many elements sequentially, for find.

Definition at line 159 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_find\_template().

#### 4.224.3.11 for\_each\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::for\_each\_minimal\_n

Minimal input size for for\_each.

Definition at line 162 of file settings.h.

#### 4.224.3.12 generate\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::generate\_minimal\_n

Minimal input size for generate.

Definition at line 165 of file settings.h.

#### 4.224.3.13 L1\_cache\_size

unsigned long long \_\_gnu\_parallel::\_Settings::L1\_cache\_size

size of the L1 cache in bytes (underestimation).

Definition at line 253 of file settings.h.

#### 4.224.3.14 L2\_cache\_size

unsigned long long \_\_gnu\_parallel::\_Settings::L2\_cache\_size

size of the L2 cache in bytes (underestimation).

Definition at line 256 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_random\_shuffle\_drs(), and \_\_gnu\_parallel::\_\_sequential\_random\_shuffle().

**4.224.3.15 max\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::max_element_minimal_n`

Minimal input size for max\_element.

Definition at line 168 of file settings.h.

**4.224.3.16 merge\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::merge_minimal_n`

Minimal input size for merge.

Definition at line 171 of file settings.h.

**4.224.3.17 merge\_oversampling**

`unsigned int __gnu_parallel::_Settings::merge_oversampling`

Oversampling factor for merge.

Definition at line 174 of file settings.h.

Referenced by `__gnu_parallel::multiway_merge_sampling_splitting()`.

**4.224.3.18 min\_element\_minimal\_n**

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::min_element_minimal_n`

Minimal input size for min\_element.

Definition at line 177 of file settings.h.

**4.224.3.19 multiway\_merge\_minimal\_k**

`int __gnu_parallel::_Settings::multiway_merge_minimal_k`

Oversampling factor for multiway\_merge.

Definition at line 183 of file settings.h.

#### 4.224.3.20 multiway\_merge\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::multiway\_merge\_minimal\_n

Minimal input size for multiway\_merge.

Definition at line 180 of file settings.h.

#### 4.224.3.21 multiway\_merge\_oversampling

unsigned int \_\_gnu\_parallel::\_Settings::multiway\_merge\_oversampling

Oversampling factor for multiway\_merge.

Definition at line 186 of file settings.h.

#### 4.224.3.22 nth\_element\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::nth\_element\_minimal\_n

Minimal input size for nth\_element.

Definition at line 189 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_nth\_element().

#### 4.224.3.23 partial\_sort\_minimal\_n

[\\_SequenceIndex](#) \_\_gnu\_parallel::\_Settings::partial\_sort\_minimal\_n

Minimal input size for partial\_sort.

Definition at line 202 of file settings.h.

#### 4.224.3.24 partial\_sum\_dilation

float \_\_gnu\_parallel::\_Settings::partial\_sum\_dilation

Ratio for partial\_sum. Assume "sum and write result" to be this factor slower than just "sum".

Definition at line 206 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear().

**4.224.3.25 partial\_sum\_minimal\_n**

```
unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n
```

Minimal input size for partial\_sum.

Definition at line 209 of file settings.h.

**4.224.3.26 partition\_chunk\_share**

```
double __gnu_parallel::_Settings::partition_chunk_share
```

Chunk size for partition, relative to input size. If > 0.0, this value overrides partition\_chunk\_size.

Definition at line 196 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partition().

**4.224.3.27 partition\_chunk\_size**

```
_SequenceIndex __gnu_parallel::_Settings::partition_chunk_size
```

Chunk size for partition.

Definition at line 192 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_partition().

**4.224.3.28 partition\_minimal\_n**

```
_SequenceIndex __gnu_parallel::_Settings::partition_minimal_n
```

Minimal input size for partition.

Definition at line 199 of file settings.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_nth\_element().

**4.224.3.29 qsb\_steals**

```
_SequenceIndex __gnu_parallel::_Settings::qsb_steals
```

The number of stolen ranges in load-balanced quicksort.

Definition at line 269 of file settings.h.



#### 4.224.3.30 random\_shuffle\_minimal\_n

`unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n`

Minimal input size for random\_shuffle.

Definition at line 212 of file settings.h.

#### 4.224.3.31 replace\_minimal\_n

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::replace_minimal_n`

Minimal input size for replace and replace\_if.

Definition at line 215 of file settings.h.

#### 4.224.3.32 search\_minimal\_n

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::search_minimal_n`

Minimal input size for search and search\_n.

Definition at line 272 of file settings.h.

#### 4.224.3.33 set\_difference\_minimal\_n

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_difference_minimal_n`

Minimal input size for set\_difference.

Definition at line 218 of file settings.h.

#### 4.224.3.34 set\_intersection\_minimal\_n

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::set_intersection_minimal_n`

Minimal input size for set\_intersection.

Definition at line 221 of file settings.h.

**4.224.3.35 set\_symmetric\_difference\_minimal\_n**

`_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`

Minimal input size for set\_symmetric\_difference.

Definition at line 224 of file settings.h.

**4.224.3.36 set\_union\_minimal\_n**

`_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`

Minimal input size for set\_union.

Definition at line 227 of file settings.h.

**4.224.3.37 sort\_minimal\_n**

`_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`

Minimal input size for parallel sorting.

Definition at line 230 of file settings.h.

**4.224.3.38 sort\_mwms\_oversampling**

`unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`

Oversampling factor for parallel std::sort (MWMS).

Definition at line 233 of file settings.h.

Referenced by `__gnu_parallel::parallel_sort_mwms()`, and `__gnu_parallel::parallel_sort_mwms_pu()`.

**4.224.3.39 sort\_qs\_num\_samples\_preset**

`unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset`

Such many samples to take to find a good pivot (quicksort).

Definition at line 236 of file settings.h.

4.224.3.40 `sort_qsb_base_case_maximal_n`

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::sort_qsb_base_case_maximal_n`

Maximal subsequence `__length` to switch to unbalanced `__base` case. Applies to `std::sort` with dynamically load-balanced quicksort.

Definition at line 240 of file `settings.h`.

Referenced by `__gnu_parallel::__qsb_local_sort_with_helping()`.

4.224.3.41 `TLB_size`

`unsigned int __gnu_parallel::_Settings::TLB_size`

size of the Translation Lookaside Buffer (underestimation).

Definition at line 259 of file `settings.h`.

Referenced by `__gnu_parallel::__parallel_random_shuffle_drs()`, and `__gnu_parallel::__sequential_random_shuffle()`.

4.224.3.42 `transform_minimal_n`

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::transform_minimal_n`

Minimal input size for parallel `std::transform`.

Definition at line 243 of file `settings.h`.

4.224.3.43 `unique_copy_minimal_n`

[\\_SequenceIndex](#) `__gnu_parallel::_Settings::unique_copy_minimal_n`

Minimal input size for `unique_copy`.

Definition at line 246 of file `settings.h`.

The documentation for this struct was generated from the following file:

- [settings.h](#)

#### 4.225 `std::_Sp_ebo_helper<_Nm,_Tp,false>` Struct Template Reference

##### Public Member Functions

- `_Sp_ebo_helper` (const `_Tp` &\_\_tp)
- `_Sp_ebo_helper` (`_Tp` &&\_\_tp)

##### Static Public Member Functions

- static `_Tp` & `_S_get` (`_Sp_ebo_helper` &\_\_eboh)

##### 4.225.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper<_Nm,_Tp,false>
```

Specialization not using EBO.

Definition at line 426 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

#### 4.226 `std::_Sp_ebo_helper<_Nm,_Tp,true>` Struct Template Reference

Inherits `_Tp`.

##### Public Member Functions

- `_Sp_ebo_helper` (const `_Tp` &\_\_tp)
- `_Sp_ebo_helper` (`_Tp` &&\_\_tp)

##### Static Public Member Functions

- static `_Tp` & `_S_get` (`_Sp_ebo_helper` &\_\_eboh)

##### 4.226.1 Detailed Description

```
template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper<_Nm,_Tp,true>
```

Specialization using EBO.

Definition at line 415 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 4.227 `__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

### 4.227.1 Detailed Description

```
template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
```

Split consistently.

Definition at line 122 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 4.228 `__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWMSSortingData](#)< \_RAIter > \*\_\_sd, \_Compare &\_\_comp, const typename [std::iterator\\_traits](#)< \_RAIter >::difference\_type \_\_num\_samples) const

### 4.228.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by sampling.

Definition at line 187 of file `multiway_mergesort.h`.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

## 4.229 `__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct` Template Reference

### Public Member Functions

- void **operator()** (const [\\_ThreadIndex](#) \_\_iam, [\\_PMWMSSortingData](#)< \_RAIter > \*\_\_sd, \_Compare &\_\_comp, const typename [std::iterator\\_traits](#)< \_RAIter >::difference\_type \_\_num\_samples) const

#### 4.229.1 Detailed Description

```
template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::__SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
```

Split by exact splitting.

Definition at line 128 of file multiway\_mergesort.h.

The documentation for this struct was generated from the following file:

- [multiway\\_mergesort.h](#)

#### 4.230 std::\_\_detail::\_\_StateSeq< \_TraitsT > Class Template Reference

##### Public Types

- typedef \_NFA< \_TraitsT > **\_RegexT**

##### Public Member Functions

- **\_StateSeq** (\_RegexT &\_\_nfa, \_StateIdT \_\_s)
- **\_StateSeq** (\_RegexT &\_\_nfa, \_StateIdT \_\_s, \_StateIdT \_\_end)
- void **\_M\_append** (\_StateIdT \_\_id)
- void **\_M\_append** (const [\\_StateSeq](#) &\_\_s)
- [\\_StateSeq](#) **\_M\_clone** ()

##### Public Attributes

- \_StateIdT **\_M\_end**
- \_RegexT & **\_M\_nfa**
- \_StateIdT **\_M\_start**

#### 4.230.1 Detailed Description

```
template<typename _TraitsT>
class std::__detail::__StateSeq< _TraitsT >
```

Describes a sequence of one or more \_State, its current start and end(s). This structure contains fragments of an NFA during construction.

Definition at line 355 of file regex\_automaton.h.

The documentation for this class was generated from the following files:

- [regex\\_automaton.h](#)
- [regex\\_automaton.tcc](#)

4.231 `__gnu_cxx::Std_pointer_impl<_Tp>` Class Template Reference

## Public Types

- typedef `_Tp` **element\_type**

## Public Member Functions

- `_Tp * get () const`
- `bool operator< (const \_Std\_pointer\_impl &__rarg) const`
- `bool operator== (const \_Std\_pointer\_impl &__rarg) const`
- `void set (element_type *__arg)`

## 4.231.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::Std_pointer_impl<_Tp>
```

A storage policy for use with `_Pointer_adapter<>` which yields a standard pointer.

A `_Storage_policy` is required to provide 4 things: 1) A `get()` API for returning the stored pointer value. 2) An `set()` API for storing a pointer value. 3) An `element_type` typedef to define the type this points to. 4) An `operator<()` to support pointer comparison. 5) An `operator==()` to support pointer comparison.

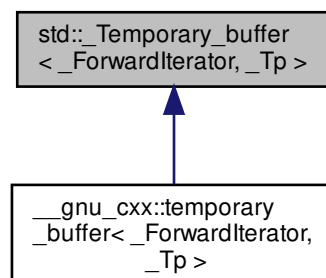
Definition at line 69 of file `pointer.h`.

The documentation for this class was generated from the following file:

- [pointer.h](#)

4.232 `std::Temporary_buffer<_ForwardIterator, _Tp>` Class Template Reference

Inheritance diagram for `std::Temporary_buffer<_ForwardIterator, _Tp>`:



### Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

### Public Member Functions

- [\\_Temporary\\_buffer](#) (\_ForwardIterator \_\_seed, size\_type \_\_original\_len)
- iterator [begin](#) ()
- iterator [end](#) ()
- size\_type [requested\\_size](#) () const
- size\_type [size](#) () const

### Protected Attributes

- pointer **\_M\_buffer**
- size\_type **\_M\_len**
- size\_type **\_M\_original\_len**

#### 4.232.1 Detailed Description

```
template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer< _ForwardIterator, _Tp >
```

This class is used in two places: `stl_algo.h` and `ext/memory`, where it is wrapped as the `temporary_buffer` class. See [temporary\\_buffer](#) docs for more notes.

Definition at line 136 of file `stl_tempbuf.h`.

#### 4.232.2 Constructor & Destructor Documentation

##### 4.232.2.1 \_Temporary\_buffer()

```
template<typename _ForwardIterator , typename _Tp >
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
 _ForwardIterator __seed,
 size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

Definition at line 258 of file `stl_tempbuf.h`.

References `std::pair< _T1, _T2 >::first`.



### 4.232.3 Member Function Documentation

#### 4.232.3.1 `begin()`

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline]
```

As per Table mumble.

Definition at line 165 of file `stl_tempbuf.h`.

#### 4.232.3.2 `end()`

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline]
```

As per Table mumble.

Definition at line 170 of file `stl_tempbuf.h`.

#### 4.232.3.3 `requested_size()`

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline]
```

Returns the size requested by the constructor; may be `>size()`.

Definition at line 160 of file `stl_tempbuf.h`.

#### 4.232.3.4 `size()`

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline]
```

As per Table mumble.

Definition at line 155 of file `stl_tempbuf.h`.

The documentation for this class was generated from the following file:

- [stl\\_tempbuf.h](#)

#### 4.233 `std::_Tuple_impl<_Idx, _Elements>` Struct Template Reference

##### 4.233.1 Detailed Description

```
template<std::size_t _Idx, typename... _Elements>
struct std::_Tuple_impl<_Idx, _Elements>
```

Contains the actual implementation of the `tuple` template, stored as a recursive inheritance hierarchy from the first element (most derived class) to the last (least derived class). The `Idx` parameter gives the 0-based index of the element stored at this point in the hierarchy; we use it to implement a constant-time `get()` operation.

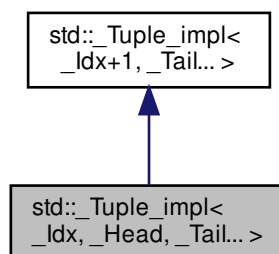
Definition at line 183 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 4.234 `std::_Tuple_impl<_Idx, _Head, _Tail...>` Struct Template Reference

Inheritance diagram for `std::_Tuple_impl<_Idx, _Head, _Tail...>`:



##### Public Types

- `typedef _Head_base<_Idx, _Head> _Base`
- `typedef \_Tuple\_impl<_Idx+1, _Tail...> _Inherited`

## Public Member Functions

- constexpr **\_Tuple\_impl** (const \_Head &\_\_head, const \_Tail &... \_\_tail)
- template<typename \_UHead, typename... \_UTail, typename = typename enable\_if<sizeof...(\_Tail) == sizeof...(\_UTail)>::type>  
constexpr **\_Tuple\_impl** (\_UHead &&\_\_head, \_UTail &&... \_\_tail)
- constexpr **\_Tuple\_impl** (const **\_Tuple\_impl** &)=default
- constexpr **\_Tuple\_impl** (**\_Tuple\_impl** &&\_\_in) noexcept(\_\_and< [is\\_nothrow\\_move\\_constructible](#)< \_Head >, [is\\_nothrow\\_move\\_constructible](#)< [\\_Inherited](#) >>::value)
- template<typename... \_UElements>  
constexpr **\_Tuple\_impl** (const **\_Tuple\_impl**< \_Idx, \_UElements... > &\_\_in)
- template<typename \_UHead, typename... \_UTails>  
constexpr **\_Tuple\_impl** (**\_Tuple\_impl**< \_Idx, \_UHead, \_UTails... > &&\_\_in)
- template<typename \_Alloc >  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a)
- template<typename \_Alloc >  
**\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, const \_Head &\_\_head, const \_Tail &... \_\_tail)
- template<typename \_Alloc, typename \_UHead, typename... \_UTail, typename = typename enable\_if<sizeof...(\_Tail) == sizeof...(\_UTail)>::type>  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, \_UHead &&\_\_head, \_UTail &&... \_\_tail)
- template<typename \_Alloc >  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, const **\_Tuple\_impl** &\_\_in)
- template<typename \_Alloc >  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, **\_Tuple\_impl** &&\_\_in)
- template<typename \_Alloc, typename \_UHead, typename... \_UTails>  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, const **\_Tuple\_impl**< \_Idx, \_UHead, \_UTails... > &&\_\_in)
- template<typename \_Alloc, typename \_UHead, typename... \_UTails>  
constexpr **\_Tuple\_impl** (allocator\_arg\_t \_\_tag, const \_Alloc &\_\_a, **\_Tuple\_impl**< \_Idx, \_UHead, \_UTails... > &&\_\_in)
- template<typename... \_UElements>  
constexpr void **\_M\_assign** (const **\_Tuple\_impl**< \_Idx, \_UElements... > &\_\_in)
- template<typename \_UHead, typename... \_UTails>  
constexpr void **\_M\_assign** (**\_Tuple\_impl**< \_Idx, \_UHead, \_UTails... > &&\_\_in)
- **\_Tuple\_impl** & operator= (const **\_Tuple\_impl** &)=delete

## Static Public Member Functions

- static constexpr \_Head & **\_M\_head** (**\_Tuple\_impl** &\_\_t) noexcept
- static constexpr const \_Head & **\_M\_head** (const **\_Tuple\_impl** &\_\_t) noexcept
- static constexpr [\\_Inherited](#) & **\_M\_tail** (**\_Tuple\_impl** &\_\_t) noexcept
- static constexpr const [\\_Inherited](#) & **\_M\_tail** (const **\_Tuple\_impl** &\_\_t) noexcept

## Protected Member Functions

- constexpr void **\_M\_swap** (**\_Tuple\_impl** &\_\_in)

## Friends

- template<std::size\_t, typename... >  
class **\_Tuple\_impl**

#### 4.234.1 Detailed Description

```
template<std::size_t _Idx, typename _Head, typename... _Tail>
struct std::_Tuple_impl<_Idx, _Head, _Tail... >
```

Recursive tuple implementation. Here we store the `Head` element and derive from a `Tuple_impl` containing the remaining elements (which contains the `Tail`).

Definition at line 191 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 4.235 `__gnu_cxx::_Unqualified_type<_Tp>` Struct Template Reference

##### Public Types

- `typedef _Tp type`

#### 4.235.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::_Unqualified_type<_Tp >
```

This structure accommodates the way in which `std::iterator_traits<>` is normally specialized for `const T*`, so that `value_of_type` is still `T`.

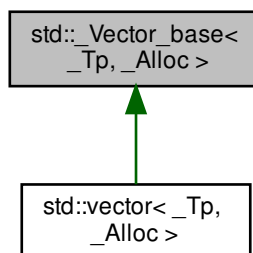
Definition at line 244 of file `pointer.h`.

The documentation for this struct was generated from the following file:

- [pointer.h](#)

#### 4.236 `std::_Vector_base<_Tp, _Alloc>` Struct Template Reference

Inheritance diagram for `std::_Vector_base<_Tp, _Alloc>`:



## Public Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Alloc >::template rebind< \_Tp >::other **\_Tp\_alloc\_type**
- typedef \_Alloc **allocator\_type**
- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)< \_Tp\_alloc\_type >::pointer **pointer**

## Public Member Functions

- **\_Vector\_base** (const allocator\_type &\_\_a) noexcept
- **\_Vector\_base** (size\_t \_\_n)
- **\_Vector\_base** (size\_t \_\_n, const allocator\_type &\_\_a)
- **\_Vector\_base** ([\\_Vector\\_base](#) &&)=default
- **\_Vector\_base** (\_Tp\_alloc\_type &&\_\_a) noexcept
- **\_Vector\_base** ([\\_Vector\\_base](#) &&\_\_x, const allocator\_type &\_\_a)
- **\_Vector\_base** (const allocator\_type &\_\_a, [\\_Vector\\_base](#) &&\_\_x)
- pointer **\_M\_allocate** (size\_t \_\_n)
- void **\_M\_deallocate** (pointer \_\_p, size\_t \_\_n)
- \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () noexcept
- const \_Tp\_alloc\_type & **\_M\_get\_Tp\_allocator** () const noexcept
- allocator\_type **get\_allocator** () const noexcept

## Public Attributes

- [\\_Vector\\_impl](#) **\_M\_impl**

## Protected Member Functions

- void **\_M\_create\_storage** (size\_t \_\_n)

## 4.236.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::_Vector_base< _Tp, _Alloc >
```

See bits/stl\_deque.h's `_Deque_base` for an explanation.

Definition at line 84 of file `stl_vector.h`.

The documentation for this struct was generated from the following file:

- [stl\\_vector.h](#)

## 4.237 std::add\_const&lt; \_Tp &gt; Struct Template Reference

## Public Types

- typedef \_Tp const **type**

#### 4.237.1 Detailed Description

```
template<typename _Tp>
struct std::add_const<_Tp >
```

add\_const

Definition at line 1544 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.238 std::add\_cv<\_Tp > Struct Template Reference

##### Public Types

- typedef [add\\_const](#)< typename [add\\_volatile](#)<\_Tp >::type >::type **type**

#### 4.238.1 Detailed Description

```
template<typename _Tp>
struct std::add_cv<_Tp >
```

add\_cv

Definition at line 1554 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.239 std::add\_lvalue\_reference<\_Tp > Struct Template Reference

Inherits std::\_\_add\_lvalue\_reference\_helper<\_Tp, bool >.

##### Public Types

- typedef \_Tp **type**

## 4.239.1 Detailed Description

```
template<typename _Tp>
struct std::add_lvalue_reference<_Tp>
```

`add_lvalue_reference`

Definition at line 1614 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.240 `std::add_rvalue_reference<_Tp>` Struct Template Reference

Inherits `std::__add_rvalue_reference_helper<_Tp, bool>`.

## Public Types

- `typedef _Tp type`

## 4.240.1 Detailed Description

```
template<typename _Tp>
struct std::add_rvalue_reference<_Tp>
```

`add_rvalue_reference`

Definition at line 1628 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.241 `std::add_volatile<_Tp>` Struct Template Reference

## Public Types

- `typedef _Tp volatile type`

#### 4.241.1 Detailed Description

```
template<typename _Tp>
struct std::add_volatile< _Tp >
```

add\_volatile

Definition at line 1549 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.242 std::adopt\_lock\_t Struct Reference

##### 4.242.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.

Definition at line 136 of file std\_mutex.h.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

#### 4.243 std::aligned\_storage< \_Len, \_Align > Struct Template Reference

##### 4.243.1 Detailed Description

```
template<std::size_t _Len, std::size_t _Align = __alignof__(typename __aligned_storage_msa<_Len>::__type)>
struct std::aligned_storage< _Len, _Align >
```

Alignment type.

The value of \_Align is a default-alignment which shall be the most stringent alignment requirement for any C++ object type whose size is no greater than \_Len (3.9). The member typedef type shall be a POD type suitable for use as uninitialized storage for any object whose size is at most \_Len and whose alignment is a divisor of \_Align.

Definition at line 2069 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 4.244 `std::aligned_union< _Len, _Types >` Struct Template Reference

### Public Types

- typedef `aligned_storage< _S_len, alignment_value >::type` `type`

### Static Public Attributes

- static const size\_t `alignment_value`

#### 4.244.1 Detailed Description

```
template<size_t _Len, typename... _Types>
struct std::aligned_union< _Len, _Types >
```

Provide aligned storage for types.

[meta.trans.other]

Provides aligned storage for any of the provided types of at least size `_Len`.

#### See also

`aligned_storage`

Definition at line 2107 of file `type_traits`.

#### 4.244.2 Member Typedef Documentation

##### 4.244.2.1 `type`

```
template<size_t _Len, typename... _Types>
typedef aligned_storage<_S_len, alignment_value>::type std::aligned_union< _Len, _Types >::type
```

The storage.

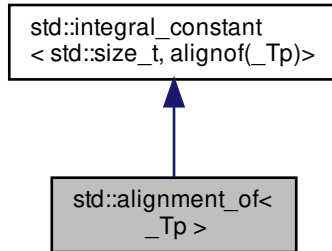
Definition at line 2119 of file `type_traits`.

The documentation for this struct was generated from the following file:

- `type_traits`

#### 4.245 `std::alignment_of< _Tp >` Struct Template Reference

Inheritance diagram for `std::alignment_of< _Tp >`:



##### Public Types

- typedef [integral\\_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr `std::size_t` **value**

##### 4.245.1 Detailed Description

```
template<typename _Tp>
struct std::alignment_of< _Tp >
```

`alignment_of`

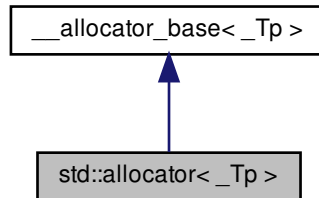
Definition at line 1350 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.246 std::allocator&lt; \_Tp &gt; Class Template Reference

Inheritance diagram for std::allocator< \_Tp >:



## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef ptrdiff\_t **difference\_type**
- typedef [true\\_type](#) **is\_always\_equal**
- typedef \_Tp \* **pointer**
- typedef [true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **allocator** (const [allocator](#) &\_\_a) noexcept
- template<typename \_Tp1 >  
constexpr **allocator** (const [allocator](#)< \_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- \_Tp \* **allocate** (size\_type \_\_n, const void \*\*=static\_cast< const void \* >(0))
- template<typename \_Up , typename... \_Args>  
void **construct** (\_Up \* \_\_p, \_Args &&... \_\_args) noexcept([std::is\\_nothrow\\_constructible](#)< \_Up, \_Args... >::value)
- void **deallocate** (\_Tp \* \_\_p, size\_type \_\_t)
- template<typename \_Up >  
void **destroy** (\_Up \* \_\_p) noexcept([std::is\\_nothrow\\_destructible](#)< \_Up >::value)
- size\_type **max\_size** () const noexcept
- [allocator](#) & **operator=** (const [allocator](#) &)=default

## Friends

- constexpr bool **operator!=** (const [allocator](#) &, const [allocator](#) &) noexcept
- constexpr bool **operator==** (const [allocator](#) &, const [allocator](#) &) noexcept

#### 4.246.1 Detailed Description

```
template<typename _Tp>
class std::allocator<_Tp>
```

The *standard* allocator, as per [20.4].

See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator> for further details.

##### Template Parameters

|                  |                           |
|------------------|---------------------------|
| <code>_Tp</code> | Type of allocated object. |
|------------------|---------------------------|

Definition at line 116 of file allocator.h.

The documentation for this class was generated from the following file:

- [allocator.h](#)

#### 4.247 `std::allocator< void >` Class Template Reference

##### Public Types

- typedef const void \* **const\_pointer**
- typedef ptrdiff\_t **difference\_type**
- typedef [true\\_type](#) **is\_always\_equal**
- typedef void \* **pointer**
- typedef [true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef size\_t **size\_type**
- typedef void **value\_type**

##### Public Member Functions

- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args) noexcept([std::is\\_nothrow\\_constructible](#)<\_Up, \_Args... >::value)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p) noexcept([std::is\\_nothrow\\_destructible](#)<\_Up >::value)

#### 4.247.1 Detailed Description

```
template<>
class std::allocator< void >
```

`allocator<void>` specialization.

Definition at line 65 of file allocator.h.

The documentation for this class was generated from the following file:

- [allocator.h](#)

## 4.248 std::allocator\_arg\_t Struct Reference

## 4.248.1 Detailed Description

[allocator.tag]

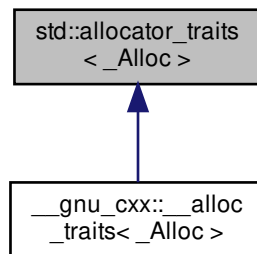
Definition at line 50 of file uses\_allocator.h.

The documentation for this struct was generated from the following file:

- uses\_allocator.h

## 4.249 std::allocator\_traits&lt;\_Alloc&gt; Struct Template Reference

Inheritance diagram for std::allocator\_traits<\_Alloc>:



## Public Types

- typedef `_Alloc` `allocator_type`
- using `const_pointer` = `typename _Ptr< __c_pointer, const value_type >::type`
- using `const_void_pointer` = `typename _Ptr< __cv_pointer, const void >::type`
- using `difference_type` = `typename _Diff< _Alloc, pointer >::type`
- using `is_always_equal` = `__detected_or_t< typename is_empty< _Alloc >::type, __equal, _Alloc >`
- using `pointer` = `__detected_or_t< value_type *, __pointer, _Alloc >`
- using `propagate_on_container_copy_assignment` = `__detected_or_t< false_type, __pocca, _Alloc >`
- using `propagate_on_container_move_assignment` = `__detected_or_t< false_type, __pocma, _Alloc >`
- using `propagate_on_container_swap` = `__detected_or_t< false_type, __pocs, _Alloc >`
- `template<typename _Tp>`  
using `rebind_alloc` = `__alloc_rebind< _Alloc, _Tp >`
- `template<typename _Tp>`  
using `rebind_traits` = `allocator_traits< rebind_alloc< _Tp > >`
- using `size_type` = `typename _Size< _Alloc, difference_type >::type`
- typedef `_Alloc::value_type` `value_type`
- using `void_pointer` = `typename _Ptr< __v_pointer, void >::type`

## Static Public Member Functions

- static constexpr [pointer allocate](#) ([\\_Alloc](#) &[\\_\\_a](#), [size\\_type](#) [\\_\\_n](#))
- static constexpr [pointer allocate](#) ([\\_Alloc](#) &[\\_\\_a](#), [size\\_type](#) [\\_\\_n](#), [const\\_void\\_pointer](#) [\\_\\_hint](#))
- template<typename [\\_Tp](#) , typename... [\\_Args](#)>  
static constexpr auto [construct](#) ([\\_Alloc](#) &[\\_\\_a](#), [\\_Tp](#) \*[\\_\\_p](#), [\\_Args](#) &&... [\\_\\_args](#)) noexcept(noexcept([\\_S\\_construct](#)([\\_\\_a](#), [\\_\\_p](#), [std::forward](#)< [\\_Args](#) >([\\_\\_args](#))...))) -> decltype([\\_S\\_construct](#)([\\_\\_a](#), [\\_\\_p](#), [std::forward](#)< [\\_Args](#) >([\\_\\_args](#))...))
- static constexpr void [deallocate](#) ([\\_Alloc](#) &[\\_\\_a](#), [pointer](#) [\\_\\_p](#), [size\\_type](#) [\\_\\_n](#))
- template<typename [\\_Tp](#) >  
static constexpr void [destroy](#) ([\\_Alloc](#) &[\\_\\_a](#), [\\_Tp](#) \*[\\_\\_p](#)) noexcept(noexcept([\\_S\\_destroy](#)([\\_\\_a](#), [\\_\\_p](#), 0)))
- static constexpr [size\\_type max\\_size](#) (const [\\_Alloc](#) &[\\_\\_a](#)) noexcept
- static constexpr [\\_Alloc select\\_on\\_container\\_copy\\_construction](#) (const [\\_Alloc](#) &[\\_\\_rhs](#))

## Protected Types

- template<typename [\\_Tp](#) >  
using [\\_\\_c\\_pointer](#) = typename [\\_Tp](#)::const\_pointer
- template<typename [\\_Tp](#) >  
using [\\_\\_cv\\_pointer](#) = typename [\\_Tp](#)::const\_void\_pointer
- template<typename [\\_Tp](#) >  
using [\\_\\_equal](#) = typename [\\_Tp](#)::is\_always\_equal
- template<typename [\\_Tp](#) >  
using [\\_\\_pocca](#) = typename [\\_Tp](#)::propagate\_on\_container\_copy\_assignment
- template<typename [\\_Tp](#) >  
using [\\_\\_pocma](#) = typename [\\_Tp](#)::propagate\_on\_container\_move\_assignment
- template<typename [\\_Tp](#) >  
using [\\_\\_pocs](#) = typename [\\_Tp](#)::propagate\_on\_container\_swap
- template<typename [\\_Tp](#) >  
using [\\_\\_pointer](#) = typename [\\_Tp](#)::pointer
- template<typename [\\_Tp](#) >  
using [\\_\\_v\\_pointer](#) = typename [\\_Tp](#)::void\_pointer

### 4.249.1 Detailed Description

```
template<typename _Alloc>
struct std::allocator_traits< _Alloc >
```

Uniform interface to all allocator types.

Definition at line 86 of file `bits/alloc_traits.h`.

### 4.249.2 Member Typedef Documentation

#### 4.249.2.1 allocator\_type

```
template<typename _Alloc>
typedef _Alloc std::allocator_traits< _Alloc >::allocator_type
```

The allocator type.

Definition at line 89 of file bits/alloc\_traits.h.

#### 4.249.2.2 const\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_pointer = typename _Ptr<__c_pointer, const value_type>↵
::type
```

The allocator's const pointer type.

Alloc::const\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<const value\_type>

Definition at line 138 of file bits/alloc\_traits.h.

#### 4.249.2.3 const\_void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer = typename _Ptr<__cv_pointer, const
void>::type
```

The allocator's const void pointer type.

Alloc::const\_void\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<const void>

Definition at line 154 of file bits/alloc\_traits.h.

#### 4.249.2.4 difference\_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type = typename _Diff<_Alloc, pointer>::type
```

The allocator's difference type.

Alloc::difference\_type if that type exists, otherwise pointer\_traits<pointer>::difference↵\_type

Definition at line 162 of file bits/alloc\_traits.h.

#### 4.249.2.5 is\_always\_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal = __detected_or_t<typename is_empty<_↵
_Alloc>::type, __equal, _Alloc>
```

Whether all instances of the allocator type compare equal.

`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

Definition at line 206 of file `bits/alloc_traits.h`.

#### 4.249.2.6 pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer = __detected_or_t<value_type*, __pointer, _Alloc>
```

The allocator's pointer type.

`Alloc::pointer` if that type exists, otherwise `value_type*`

Definition at line 98 of file `bits/alloc_traits.h`.

#### 4.249.2.7 propagate\_on\_container\_copy\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment = __detected_or_↵
t<false_type, __pocca, _Alloc>
```

How the allocator is propagated on copy assignment.

`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

Definition at line 179 of file `bits/alloc_traits.h`.

#### 4.249.2.8 propagate\_on\_container\_move\_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment = __detected_or_↵
t<false_type, __pocma, _Alloc>
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

Definition at line 188 of file `bits/alloc_traits.h`.



#### 4.249.2.9 propagate\_on\_container\_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap = __detected_or_t<false_type,
__pocs, _Alloc>
```

How the allocator is propagated on swap.

Alloc::propagate\_on\_container\_swap if that type exists, otherwise false\_type

Definition at line 197 of file bits/alloc\_traits.h.

#### 4.249.2.10 size\_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type = typename _Size<_Alloc, difference_type>::type
```

The allocator's size type.

Alloc::size\_type if that type exists, otherwise make\_unsigned<difference\_type>::type

Definition at line 170 of file bits/alloc\_traits.h.

#### 4.249.2.11 value\_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type
```

The allocated type.

Definition at line 91 of file bits/alloc\_traits.h.

#### 4.249.2.12 void\_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer = typename _Ptr<__v_pointer, void>::type
```

The allocator's void pointer type.

Alloc::void\_pointer if that type exists, otherwise pointer\_traits<pointer>::rebind<void>

Definition at line 146 of file bits/alloc\_traits.h.

### 4.249.3 Member Function Documentation

#### 4.249.3.1 allocate() [1/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n) [inline], [static]
```

Allocate memory.

**Parameters**

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

Definition at line 313 of file `bits/alloc_traits.h`.

Referenced by `std::__allocate_guarded()`.

**4.249.3.2 allocate()** [2/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
 _Alloc & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static]
```

Allocate memory.

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

**Returns**

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

Definition at line 328 of file `bits/alloc_traits.h`.

**4.249.3.3 construct()**

```
template<typename _Alloc>
template<typename _Tp , typename... _Args>
static constexpr auto std::allocator_traits< _Alloc >::construct (
 _Alloc & __a,
 _Tp * __p,
 _Args &&... __args) -> decltype(_S_construct(__a, __p, std::forward<_Args>(__args)...))
 [inline], [static], [noexcept]
```

Construct an object of type `_Tp`.

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                        |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for <code>Tp</code> |
| <code>__args</code> | Constructor arguments.                                               |

Calls `__a.construct(__p, std::forward<Args>(__args)...) if that expression is well-formed, otherwise uses placement-new to construct an object of type _Tp at location __p from the arguments __args...`

Definition at line 356 of file `bits/alloc_traits.h`.

## 4.249.3.4 deallocate()

```
template<typename _Alloc>
static constexpr void std::allocator_traits<_Alloc>::deallocate (
 _Alloc & __a,
 pointer __p,
 size_type __n) [inline], [static]
```

Deallocate memory.

## Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__a</code> | An allocator.                                  |
| <code>__p</code> | Pointer to the memory to deallocate.           |
| <code>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

Definition at line 340 of file `bits/alloc_traits.h`.

Referenced by `std::__allocated_ptr<_Alloc>::~~__allocated_ptr()`.

## 4.249.3.5 destroy()

```
template<typename _Alloc>
template<typename _Tp>
static constexpr void std::allocator_traits<_Alloc>::destroy (
 _Alloc & __a,
 _Tp * __p) [inline], [static], [noexcept]
```

Destroy an object of type `_Tp`.

**Parameters**

|       |                                  |
|-------|----------------------------------|
| $\_a$ | An allocator.                    |
| $\_p$ | Pointer to the object to destroy |

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~Tp()`

Definition at line 372 of file `bits/alloc_traits.h`.

Referenced by `std::_Destroy()`.

**4.249.3.6 max\_size()**

```
template<typename _Alloc>
static constexpr size_type std::allocator_traits< _Alloc >::max_size (
 const _Alloc & __a) [inline], [static], [noexcept]
```

The maximum supported allocation size.

**Parameters**

|       |               |
|-------|---------------|
| $\_a$ | An allocator. |
|-------|---------------|

**Returns**

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_type>::max()`

Definition at line 385 of file `bits/alloc_traits.h`.

Referenced by `std::forward_list<_Tp, _Alloc>::max_size()`, and `std::list<__inp, __rebind_inp>::max_size()`.

**4.249.3.7 select\_on\_container\_copy\_construction()**

```
template<typename _Alloc>
static constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
 const _Alloc & __rhs) [inline], [static]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

## Returns

`__rhs.select_on_container_copy_construction()` or `__rhs`

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns `__rhs`

Definition at line 397 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

4.250 `std::allocator_traits< allocator< _Tp > >` Struct Template Reference

## Public Types

- using `allocator_type` = `allocator< _Tp >`
- using `const_pointer` = `const _Tp *`
- using `const_void_pointer` = `const void *`
- using `difference_type` = `std::ptrdiff_t`
- using `is_always_equal` = `true_type`
- using `pointer` = `_Tp *`
- using `propagate_on_container_copy_assignment` = `false_type`
- using `propagate_on_container_move_assignment` = `true_type`
- using `propagate_on_container_swap` = `false_type`
- template<typename `_Up` >  
using `rebind_alloc` = `allocator< _Up >`
- template<typename `_Up` >  
using `rebind_traits` = `allocator_traits< allocator< _Up > >`
- using `size_type` = `std::size_t`
- using `value_type` = `_Tp`
- using `void_pointer` = `void *`

## Static Public Member Functions

- static constexpr `pointer` `allocate` (`allocator_type` &`__a`, `size_type` `__n`)
- static constexpr `pointer` `allocate` (`allocator_type` &`__a`, `size_type` `__n`, `const_void_pointer` `__hint`)
- template<typename `_Up`, typename... `_Args`>  
static constexpr void `construct` (`allocator_type` &`__a`, `_Up *``__p`, `_Args` &&... `__args`) noexcept(`std::is_nothrow_constructible< _Up, _Args... >::value`)
- static constexpr void `deallocate` (`allocator_type` &`__a`, `pointer` `__p`, `size_type` `__n`)
- template<typename `_Up` >  
static constexpr void `destroy` (`allocator_type` &`__a`, `_Up *``__p`) noexcept(`is_nothrow_destructible< _Up >::value`)
- static constexpr `size_type` `max_size` (const `allocator_type` &`__a`) noexcept
- static constexpr `allocator_type` `select_on_container_copy_construction` (const `allocator_type` &`__rhs`)

#### 4.250.1 Detailed Description

```
template<typename _Tp>
struct std::allocator_traits< allocator< _Tp > >
```

Partial specialization for std::allocator.

Definition at line 407 of file bits/alloc\_traits.h.

#### 4.250.2 Member Typedef Documentation

##### 4.250.2.1 allocator\_type

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::allocator_type = allocator<_Tp>
```

The allocator type.

Definition at line 410 of file bits/alloc\_traits.h.

##### 4.250.2.2 const\_pointer

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::const_pointer = const _Tp*
```

The allocator's const pointer type.

Definition at line 419 of file bits/alloc\_traits.h.

##### 4.250.2.3 const\_void\_pointer

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::const_void_pointer = const void*
```

The allocator's const void pointer type.

Definition at line 425 of file bits/alloc\_traits.h.

#### 4.250.2.4 `difference_type`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::difference_type = std::ptrdiff_t
```

The allocator's difference type.

Definition at line 428 of file `bits/alloc_traits.h`.

#### 4.250.2.5 `is_always_equal`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::is_always_equal = true_type
```

Whether all instances of the allocator type compare equal.

Definition at line 443 of file `bits/alloc_traits.h`.

#### 4.250.2.6 `pointer`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::pointer = _Tp*
```

The allocator's pointer type.

Definition at line 416 of file `bits/alloc_traits.h`.

#### 4.250.2.7 `propagate_on_container_copy_assignment`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment = false_type
```

How the allocator is propagated on copy assignment.

Definition at line 434 of file `bits/alloc_traits.h`.

#### 4.250.2.8 `propagate_on_container_move_assignment`

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment = true_type
```

How the allocator is propagated on move assignment.

Definition at line 437 of file `bits/alloc_traits.h`.

#### 4.250.2.9 propagate\_on\_container\_swap

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap = false_type
```

How the allocator is propagated on swap.

Definition at line 440 of file bits/alloc\_traits.h.

#### 4.250.2.10 size\_type

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::size_type = std::size_t
```

The allocator's size type.

Definition at line 431 of file bits/alloc\_traits.h.

#### 4.250.2.11 value\_type

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::value_type = _Tp
```

The allocated type.

Definition at line 413 of file bits/alloc\_traits.h.

#### 4.250.2.12 void\_pointer

```
template<typename _Tp >
using std::allocator_traits< allocator< _Tp > >::void_pointer = void*
```

The allocator's void pointer type.

Definition at line 422 of file bits/alloc\_traits.h.

### 4.250.3 Member Function Documentation

#### 4.250.3.1 allocate() [1/2]

```
template<typename _Tp >
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
 allocator_type & __a,
 size_type __n) [inline], [static]
```

Allocate memory.



## Parameters

|                  |                                              |
|------------------|----------------------------------------------|
| <code>__a</code> | An allocator.                                |
| <code>__n</code> | The number of objects to allocate space for. |

Calls `a.allocate(n)`

Definition at line 459 of file `bits/alloc_traits.h`.

4.250.3.2 `allocate()` [2/2]

```
template<typename _Tp >
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
 allocator_type & __a,
 size_type __n,
 const_void_pointer __hint) [inline], [static]
```

Allocate memory.

## Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__a</code>    | An allocator.                                |
| <code>__n</code>    | The number of objects to allocate space for. |
| <code>__hint</code> | Aid to locality.                             |

## Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

Definition at line 473 of file `bits/alloc_traits.h`.

4.250.3.3 `construct()`

```
template<typename _Tp >
template<typename _Up , typename... _Args>
static constexpr void std::allocator_traits< allocator< _Tp > >::construct (
 allocator_type & __a,
 _Up * __p,
 _Args &&... __args) [inline], [static], [noexcept]
```

Construct an object of type `_Up`

## Parameters

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| <code>__a</code>    | An allocator.                                                                             |
| <code>__p</code>    | Pointer to memory of suitable size and alignment for an object of type <code>_Up</code> . |
| <code>__args</code> | Constructor arguments.                                                                    |

Calls `__a.construct(__p, std::forward<_Args>(__args)...) in C++11, C++14 and C++17. Changed in C++20 to call std::construct_at(__p, std::forward<_Args>(__args)...) instead.`

Definition at line 507 of file `bits/alloc_traits.h`.

4.250.3.4 `deallocate()`

```
template<typename _Tp >
static constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
 allocator_type & __a,
 pointer __p,
 size_type __n) [inline], [static]
```

Deallocate memory.

## Parameters

|                          |                                                |
|--------------------------|------------------------------------------------|
| <code>__↵<br/>__a</code> | An allocator.                                  |
| <code>__↵<br/>__p</code> | Pointer to the memory to deallocate.           |
| <code>__↵<br/>__n</code> | The number of objects space was allocated for. |

Calls `a.deallocate(p, n)`

Definition at line 491 of file `bits/alloc_traits.h`.

4.250.3.5 `destroy()`

```
template<typename _Tp >
template<typename _Up >
static constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
 allocator_type & __a,
 _Up * __p) [inline], [static], [noexcept]
```

Destroy an object of type `_Up`.

## Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__a</code> | An allocator.                    |
| <code>__p</code> | Pointer to the object to destroy |

Calls `__a.destroy(__p)`.

Definition at line 527 of file `bits/alloc_traits.h`.

4.250.3.6 `max_size()`

```
template<typename _Tp >
static constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
 const allocator_type & __a) [inline], [static], [noexcept]
```

The maximum supported allocation size.

## Parameters

|                  |               |
|------------------|---------------|
| <code>__a</code> | An allocator. |
|------------------|---------------|

## Returns

`__a.max_size()`

Definition at line 543 of file `bits/alloc_traits.h`.

4.250.3.7 `select_on_container_copy_construction()`

```
template<typename _Tp >
static constexpr allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container←
_copy_construction (
 const allocator_type & __rhs) [inline], [static]
```

Obtain an allocator to use when copying a container.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__rhs</code> | An allocator. |
|--------------------|---------------|

**Returns**

`__rhs`

Definition at line 558 of file `bits/alloc_traits.h`.

The documentation for this struct was generated from the following file:

- [bits/alloc\\_traits.h](#)

**4.251 `__gnu_cxx::limit_condition::always_adjustor` Struct Reference**

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

**4.251.1 Detailed Description**

Always enter the condition.

Definition at line 452 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

**4.252 `__gnu_cxx::random_condition::always_adjustor` Struct Reference**

Inherits `__gnu_cxx::random_condition::adjustor_base`.

**4.252.1 Detailed Description**

Always enter the condition.

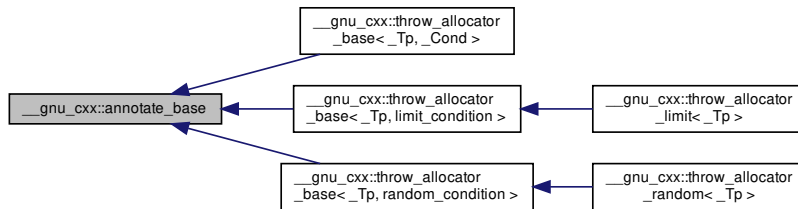
Definition at line 533 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.253 `__gnu_cxx::annotate_base` Struct Reference

Inheritance diagram for `__gnu_cxx::annotate_base`:



## Public Member Functions

- void **check** (size\_t label)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)

## Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

## Friends

- `std::ostream` & **operator**<< (`std::ostream` &, const `annotate_base` &)

## 4.253.1 Detailed Description

Base class for checking address and label information about allocations. Create a `std::map` between the allocated address (void\*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.

Definition at line 93 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.254 std::experimental::fundamentals\_v1::any Class Reference

### Public Member Functions

- [any](#) () noexcept
- [any](#) (const [any](#) &\_\_other)
- [any](#) ([any](#) &&\_\_other) noexcept
- template<typename \_ValueType , typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename enable\_if< is\_constructible< \_Tp, \_ValueType && >::value, bool >::type = true>  
[any](#) (\_ValueType &&\_\_value)
- template<typename \_ValueType , typename \_Tp = \_Decay<\_ValueType>, typename \_Mgr = \_Manager<\_Tp>, typename enable\_if< !is\_constructible< \_Tp, \_ValueType && >::value, bool >::type = false>  
[any](#) (\_ValueType &&\_\_value)
- [~any](#) ()
- void [clear](#) () noexcept
- bool [empty](#) () const noexcept
- [any](#) & [operator=](#) (const [any](#) &\_\_rhs)
- [any](#) & [operator=](#) ([any](#) &&\_\_rhs) noexcept
- template<typename \_ValueType >  
[enable\\_if\\_t](#)<!is\_same< [any](#), decay\_t< \_ValueType > >::value, [any](#) & > [operator=](#) (\_ValueType &&\_\_rhs)
- void [swap](#) ([any](#) &\_\_rhs) noexcept
- const [type\\_info](#) & [type](#) () const noexcept

### Static Public Member Functions

- template<typename \_Tp >  
static constexpr bool [\\_\\_is\\_valid\\_cast](#) ()

### Friends

- template<typename \_Tp >  
[enable\\_if\\_t](#)< [is\\_object](#)< \_Tp >::value, void \* > [\\_\\_any\\_caster](#) (const [any](#) \*\_\_any)

#### 4.254.1 Detailed Description

A type-safe container of any type.

An [any](#) object's state is either empty or it stores a contained object of CopyConstructible type.

Definition at line 90 of file experimental/any.

#### 4.254.2 Constructor & Destructor Documentation

#### 4.254.2.1 any() [1/5]

```
std::experimental::fundamentals_v1::any::any () [inline], [noexcept]
```

Default constructor, creates an empty object.

Definition at line 129 of file experimental/any.

#### 4.254.2.2 any() [2/5]

```
std::experimental::fundamentals_v1::any::any (
 const any & __other) [inline]
```

Copy constructor, copies the state of \_\_other.

Definition at line 132 of file experimental/any.

#### 4.254.2.3 any() [3/5]

```
std::experimental::fundamentals_v1::any::any (
 any && __other) [inline], [noexcept]
```

Move constructor, transfer the state from \_\_other.

#### Postcondition

\_\_other.empty() (this postcondition is a GNU extension)

Definition at line 149 of file experimental/any.

#### 4.254.2.4 any() [4/5]

```
template<typename _ValueType , typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
std::experimental::fundamentals_v1::any::any (
 _ValueType && __value) [inline]
```

Construct with a copy of \_\_value as the contained object.

Definition at line 166 of file experimental/any.

#### 4.254.2.5 any() [5/5]

```
template<typename _ValueType , typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
std::experimental::fundamentals_v1::any::any (
 _ValueType && __value) [inline]
```

Construct with a copy of `__value` as the contained object.

Definition at line 179 of file `experimental/any`.

#### 4.254.2.6 ~any()

```
std::experimental::fundamentals_v1::any::~~any () [inline]
```

Destructor, calls `clear()`

Definition at line 188 of file `experimental/any`.

### 4.254.3 Member Function Documentation

#### 4.254.3.1 clear()

```
void std::experimental::fundamentals_v1::any::clear () [inline], [noexcept]
```

If not empty, destroy the contained object.

Definition at line 230 of file `experimental/any`.

#### 4.254.3.2 empty()

```
bool std::experimental::fundamentals_v1::any::empty () const [inline], [noexcept]
```

Reports whether there is a contained object or not.

Definition at line 272 of file `experimental/any`.



**4.254.3.3 operator=()** [1/3]

```
any& std::experimental::fundamentals_v1::any::operator= (
 const any & __rhs) [inline]
```

Copy the state of another object.

Definition at line 193 of file experimental/any.

**4.254.3.4 operator=()** [2/3]

```
any& std::experimental::fundamentals_v1::any::operator= (
 any && __rhs) [inline], [noexcept]
```

Move assignment operator.

**Postcondition**

`__rhs.empty()` (not guaranteed for other implementations)

Definition at line 204 of file experimental/any.

**4.254.3.5 operator=()** [3/3]

```
template<typename _ValueType >
enable_if_t<!is_same<any, decay_t<_ValueType> >::value, any&> std::experimental::fundamentals_v1::any::operator= (
 _ValueType && __rhs) [inline]
```

Store a copy of `__rhs` as the contained object.

Definition at line 221 of file experimental/any.

**4.254.3.6 swap()**

```
void std::experimental::fundamentals_v1::any::swap (
 any & __rhs) [inline], [noexcept]
```

Exchange state with another object.

Definition at line 240 of file experimental/any.

#### 4.254.3.7 type()

```
const type_info& std::experimental::fundamentals_v1::any::type () const [inline], [noexcept]
```

The typeid of the contained object, or typeid(void) if empty.

Definition at line 276 of file experimental/any.

The documentation for this class was generated from the following file:

- [experimental/any](#)

### 4.255 std::array< \_Tp, \_Nm > Struct Template Reference

#### Public Types

- typedef \_\_array\_traits< \_Tp, \_Nm > **\_AT\_Type**
- typedef const value\_type \* **const\_iterator**
- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef [std::reverse\\_iterator](#)< const\_iterator > **const\_reverse\_iterator**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **iterator**
- typedef value\_type \* **pointer**
- typedef value\_type & **reference**
- typedef [std::reverse\\_iterator](#)< iterator > **reverse\_iterator**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr reference **at** (size\_type \_\_n)
- constexpr const\_reference **at** (size\_type \_\_n) const
- constexpr reference **back** () noexcept
- constexpr const\_reference **back** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- constexpr [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr pointer **data** () noexcept
- constexpr const\_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr void **fill** (const value\_type &\_\_u)
- constexpr reference **front** () noexcept

- constexpr const\_reference **front** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- constexpr reference **operator[]** (size\_type \_\_n) noexcept
- constexpr const\_reference **operator[]** (size\_type \_\_n) const noexcept
- constexpr [reverse\\_iterator](#) **rbegin** () noexcept
- constexpr [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr [reverse\\_iterator](#) **rend** () noexcept
- constexpr [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr void **swap** ([array](#) &\_\_other) noexcept(\_AT\_Type::\_Is\_nothrow\_swappable::value)

#### Public Attributes

- `_AT_Type::_Type` **\_M\_elems**

#### 4.255.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::array<_Tp, _Nm >
```

A standard container for storing a fixed size sequence of elements.

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#).

Sets support random access iterators.

#### Template Parameters

|                        |                                                  |
|------------------------|--------------------------------------------------|
| <i><code>Tp</code></i> | Type of element. Required to be a complete type. |
| <i><code>Nm</code></i> | Number of elements.                              |

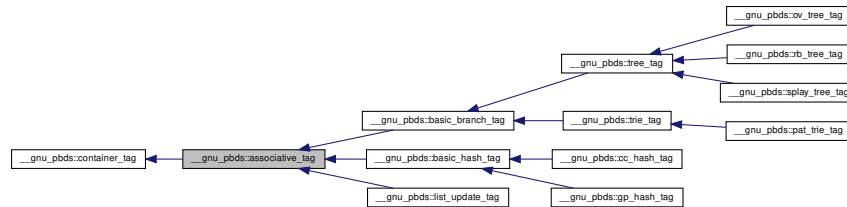
Definition at line 94 of file `array`.

The documentation for this struct was generated from the following file:

- [array](#)

## 4.256 \_\_gnu\_pbds::associative\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::associative\_tag:



### 4.256.1 Detailed Description

Basic associative-container.

Definition at line 135 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.257 std::atomic< \_Tp > Struct Template Reference

### Public Types

- using **value\_type** = \_Tp

### Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_Tp \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_s, [memory\\_order](#) \_\_f) volatile noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_Tp &\_\_e, \_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_Tp **exchange** (\_Tp \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

- `bool is_lock_free ()` const noexcept
- `bool is_lock_free ()` const volatile noexcept
- `_Tp load (memory_order __m=memory_order_seq_cst)` const noexcept
- `_Tp load (memory_order __m=memory_order_seq_cst)` const volatile noexcept
- `operator _Tp ()` const noexcept
- `operator _Tp ()` const volatile noexcept
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `_Tp operator= (_Tp __i)` noexcept
- `_Tp operator= (_Tp __i)` volatile noexcept
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst)` noexcept
- `void store (_Tp __i, memory_order __m=memory_order_seq_cst)` volatile noexcept

#### 4.257.1 Detailed Description

```
template<typename _Tp>
struct std::atomic<_Tp>
```

Generic atomic type, primary class template.

#### Template Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>_Tp</code> | Type to be made atomic, must be trivially copyable. |
|------------------|-----------------------------------------------------|

Definition at line 57 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

## 4.258 `std::atomic<_Tp*>` Struct Template Reference

### Public Types

- `typedef __atomic_base<_Tp*> __base_type`
- `typedef _Tp* __pointer_type`
- `using difference_type = ptrdiff_t`
- `using value_type = _Tp*`

### Public Member Functions

- `atomic (const atomic &)=delete`
- `constexpr atomic (__pointer_type __p)` noexcept
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2)` noexcept

- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_strong (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept`
- `bool compare_exchange_weak (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type exchange (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_add (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept`
- `__pointer_type fetch_sub (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept`
- `bool is_lock_free () const noexcept`
- `bool is_lock_free () const volatile noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const noexcept`
- `__pointer_type load (memory_order __m=memory_order_seq_cst) const volatile noexcept`
- `operator __pointer_type () const noexcept`
- `operator __pointer_type () const volatile noexcept`
- `__pointer_type operator++ (int) noexcept`
- `__pointer_type operator++ (int) volatile noexcept`
- `__pointer_type operator++ () noexcept`
- `__pointer_type operator++ () volatile noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) noexcept`
- `__pointer_type operator+= (ptrdiff_t __d) volatile noexcept`
- `__pointer_type operator-- (int) noexcept`
- `__pointer_type operator-- (int) volatile noexcept`
- `__pointer_type operator-- () noexcept`
- `__pointer_type operator-- () volatile noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) noexcept`
- `__pointer_type operator-= (ptrdiff_t __d) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__pointer_type operator= (__pointer_type __p) noexcept`
- `__pointer_type operator= (__pointer_type __p) volatile noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### Public Attributes

- `__base_type _M_b`

## 4.258.1 Detailed Description

```
template<typename _Tp>
struct std::atomic< _Tp * >
```

Partial specialization for pointer types.

Definition at line 367 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

4.259 `std::atomic< bool >` Struct Template Reference

## Public Types

- using **value\_type** = `bool`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (bool \_\_i) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (bool &\_\_i1, bool \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **exchange** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- bool **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator bool** () const noexcept
- **operator bool** () const volatile noexcept
- [atomic](#) & **operator=** (const [atomic](#) &)=delete
- [atomic](#) & **operator=** (const [atomic](#) &) volatile=delete
- bool **operator=** (bool \_\_i) noexcept
- bool **operator=** (bool \_\_i) volatile noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- void **store** (bool \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept

#### 4.259.1 Detailed Description

```
template<>
struct std::atomic< bool >
```

atomic<bool>

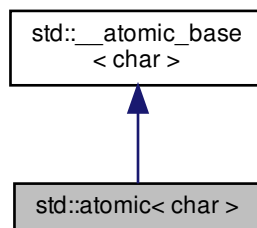
Definition at line 62 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.260 std::atomic< char > Struct Template Reference

Inheritance diagram for std::atomic< char >:



#### Public Types

- typedef [\\_\\_atomic\\_base](#)< char > **\_\_base\_type**
- typedef char **\_\_integral\_type**
- using **difference\_type** = value\_type
- using **value\_type** = char



## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.260.1 Detailed Description

```
template<>
struct std::atomic< char >
```

Explicit specialization for char.

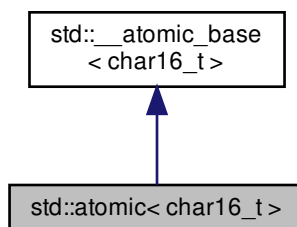
Definition at line 644 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.261 std::atomic< char16\_t > Struct Template Reference

Inheritance diagram for `std::atomic< char16_t >`:



#### Public Types

- `typedef __atomic_base< char16_t > __base_type`
- `typedef char16_t __integral_type`
- `using difference_type = value_type`
- `using value_type = char16_t`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.261.1 Detailed Description

```
template<>
struct std::atomic< char16_t >
```

Explicit specialization for `char16_t`.

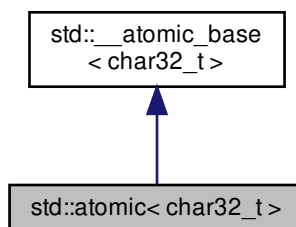
Definition at line 945 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.262 std::atomic< char32\_t > Struct Template Reference

Inheritance diagram for `std::atomic< char32_t >`:



#### Public Types

- `typedef __atomic_base< char32_t > __base_type`
- `typedef char32_t __integral_type`
- `using difference_type = value_type`
- `using value_type = char32_t`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.262.1 Detailed Description

```
template<>
struct std::atomic< char32_t >
```

Explicit specialization for `char32_t`.

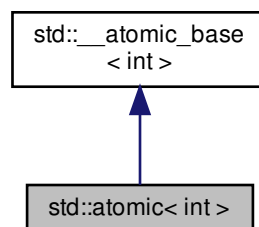
Definition at line 968 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.263 std::atomic< int > Struct Template Reference

Inheritance diagram for `std::atomic< int >`:



#### Public Types

- `typedef __atomic_base< int > __base_type`
- `typedef int __integral_type`
- `using difference_type = value_type`
- `using value_type = int`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.263.1 Detailed Description

```
template<>
struct std::atomic< int >
```

Explicit specialization for int.

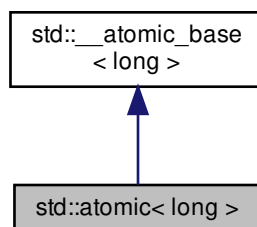
Definition at line 759 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.264 std::atomic< long > Struct Template Reference

Inheritance diagram for std::atomic< long >:



#### Public Types

- `typedef __atomic_base< long > __base_type`
- `typedef long __integral_type`
- `using difference_type = value_type`
- `using value_type = long`



## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.264.1 Detailed Description

```
template<>
struct std::atomic< long >
```

Explicit specialization for long.

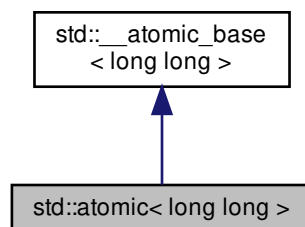
Definition at line 805 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.265 std::atomic< long long > Struct Template Reference

Inheritance diagram for std::atomic< long long >:



#### Public Types

- `typedef __atomic_base< long long > __base_type`
- `typedef long long __integral_type`
- `using difference_type = value_type`
- `using value_type = long long`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.265.1 Detailed Description

```
template<>
struct std::atomic< long long >
```

Explicit specialization for long long.

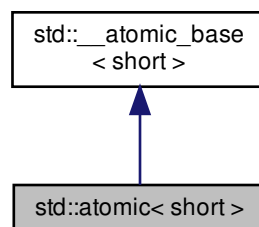
Definition at line 851 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.266 std::atomic< short > Struct Template Reference

Inheritance diagram for std::atomic< short >:



#### Public Types

- `typedef __atomic_base< short > __base_type`
- `typedef short __integral_type`
- `using difference_type = value_type`
- `using value_type = short`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.266.1 Detailed Description

```
template<>
struct std::atomic< short >
```

Explicit specialization for short.

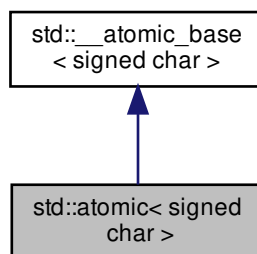
Definition at line 713 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.267 std::atomic< signed char > Struct Template Reference

Inheritance diagram for `std::atomic< signed char >`:



#### Public Types

- `typedef __atomic_base< signed char > __base_type`
- `typedef signed char __integral_type`
- `using difference_type = value_type`
- `using value_type = signed char`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.267.1 Detailed Description

```
template<>
struct std::atomic< signed char >
```

Explicit specialization for signed char.

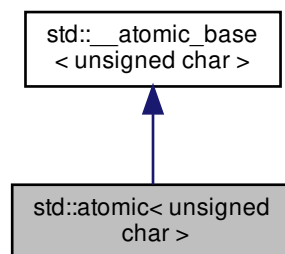
Definition at line 667 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.268 std::atomic< unsigned char > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned char >`:



#### Public Types

- `typedef __atomic_base< unsigned char > __base_type`
- `typedef unsigned char __integral_type`
- `using difference_type = value_type`
- `using value_type = unsigned char`



## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.268.1 Detailed Description

```
template<>
struct std::atomic< unsigned char >
```

Explicit specialization for unsigned char.

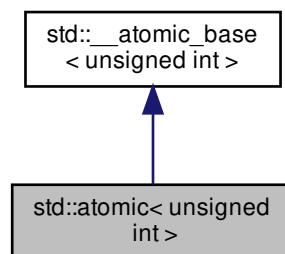
Definition at line 690 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.269 std::atomic< unsigned int > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned int >`:



#### Public Types

- `typedef __atomic_base< unsigned int > __base_type`
- `typedef unsigned int __integral_type`
- `using difference_type = value_type`
- `using value_type = unsigned int`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.269.1 Detailed Description

```
template<>
struct std::atomic< unsigned int >
```

Explicit specialization for unsigned int.

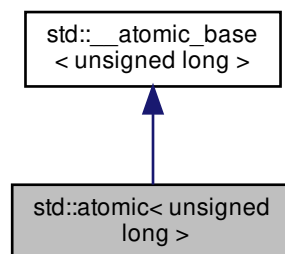
Definition at line 782 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.270 std::atomic< unsigned long > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned long >`:



#### Public Types

- `typedef __atomic_base< unsigned long > __base_type`
- `typedef unsigned long __integral_type`
- `using difference_type = value_type`
- `using value_type = unsigned long`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.270.1 Detailed Description

```
template<>
struct std::atomic< unsigned long >
```

Explicit specialization for unsigned long.

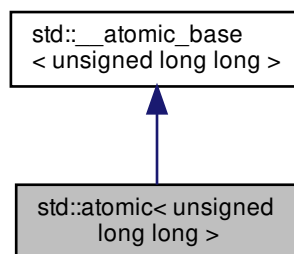
Definition at line 828 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.271 std::atomic< unsigned long long > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned long long >`:



#### Public Types

- `typedef __atomic_base< unsigned long long > __base_type`
- `typedef unsigned long long __integral_type`
- using `difference_type` = `value_type`
- using `value_type` = `unsigned long long`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.271.1 Detailed Description

```
template<>
struct std::atomic< unsigned long long >
```

Explicit specialization for unsigned long long.

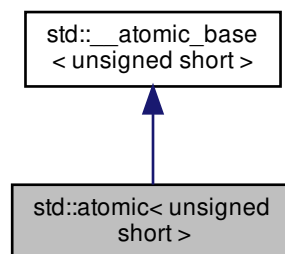
Definition at line 874 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.272 std::atomic< unsigned short > Struct Template Reference

Inheritance diagram for `std::atomic< unsigned short >`:



#### Public Types

- `typedef __atomic_base< unsigned short > __base_type`
- `typedef unsigned short __integral_type`
- `using difference_type = value_type`
- `using value_type = unsigned short`



## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.272.1 Detailed Description

```
template<>
struct std::atomic< unsigned short >
```

Explicit specialization for unsigned short.

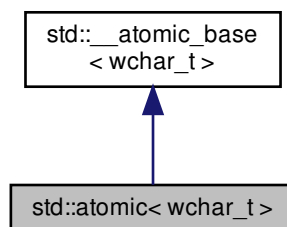
Definition at line 736 of file atomic.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.273 std::atomic< wchar\_t > Struct Template Reference

Inheritance diagram for `std::atomic< wchar_t >`:



#### Public Types

- `typedef __atomic_base< wchar_t > __base_type`
- `typedef wchar_t __integral_type`
- `using difference_type = value_type`
- `using value_type = wchar_t`

## Public Member Functions

- **atomic** (const [atomic](#) &)=delete
- constexpr **atomic** (\_\_integral\_type \_\_i) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_strong** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m1, [memory\\_order](#) \_\_m2) volatile noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- bool **compare\_exchange\_weak** (\_\_int\_type &\_\_i1, \_\_int\_type \_\_i2, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **exchange** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_add** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_and** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_or** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_sub** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) noexcept
- \_\_int\_type **fetch\_xor** (\_\_int\_type \_\_i, [memory\\_order](#) \_\_m=memory\_order\_seq\_cst) volatile noexcept
- bool **is\_lock\_free** () const noexcept
- bool **is\_lock\_free** () const volatile noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const noexcept
- \_\_int\_type **load** ([memory\\_order](#) \_\_m=memory\_order\_seq\_cst) const volatile noexcept
- **operator \_\_int\_type** () const noexcept
- **operator \_\_int\_type** () const volatile noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator&=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator++** (int) noexcept
- \_\_int\_type **operator++** (int) volatile noexcept
- \_\_int\_type **operator++** () noexcept
- \_\_int\_type **operator++** () volatile noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) noexcept
- \_\_int\_type **operator+=** (\_\_int\_type \_\_i) volatile noexcept
- \_\_int\_type **operator--** (int) noexcept
- \_\_int\_type **operator--** (int) volatile noexcept
- \_\_int\_type **operator--** () noexcept
- \_\_int\_type **operator--** () volatile noexcept
- \_\_int\_type **operator-=** (\_\_int\_type \_\_i) noexcept

- `__int_type operator-= (__int_type __i) volatile noexcept`
- `atomic & operator= (const atomic &)=delete`
- `atomic & operator= (const atomic &) volatile=delete`
- `__int_type operator^= (__int_type __i) noexcept`
- `__int_type operator^= (__int_type __i) volatile noexcept`
- `__int_type operator|= (__int_type __i) noexcept`
- `__int_type operator|= (__int_type __i) volatile noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) noexcept`
- `void store (__int_type __i, memory_order __m=memory_order_seq_cst) volatile noexcept`

#### 4.273.1 Detailed Description

```
template<>
struct std::atomic< wchar_t >
```

Explicit specialization for `wchar_t`.

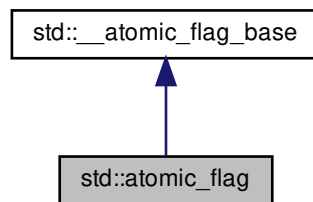
Definition at line 897 of file `atomic`.

The documentation for this struct was generated from the following file:

- [atomic](#)

#### 4.274 std::atomic\_flag Struct Reference

Inheritance diagram for `std::atomic_flag`:



#### Public Member Functions

- `atomic_flag (const atomic_flag &)=delete`
- `constexpr atomic_flag (bool __i) noexcept`
- `void clear (memory_order __m=memory_order_seq_cst) noexcept`
- `void clear (memory_order __m=memory_order_seq_cst) volatile noexcept`
- `atomic_flag & operator= (const atomic_flag &)=delete`
- `atomic_flag & operator= (const atomic_flag &) volatile=delete`
- `bool test_and_set (memory_order __m=memory_order_seq_cst) noexcept`
- `bool test_and_set (memory_order __m=memory_order_seq_cst) volatile noexcept`

## Public Attributes

- `__atomic_flag_data_type _M_i`

## 4.274.1 Detailed Description

`atomic_flag`

Definition at line 186 of file `atomic_base.h`.

The documentation for this struct was generated from the following file:

- [atomic\\_base.h](#)

4.275 `std::auto_ptr<_Tp>` Class Template Reference

## Public Types

- `typedef _Tp element\_type`

## Public Member Functions

- `auto\_ptr (element\_type *__p=0) throw ()`
- `auto\_ptr (auto\_ptr &__a) throw ()`
- `template<typename _Tp1 >  
auto\_ptr (auto\_ptr<_Tp1 > &__a) throw ()`
- `auto\_ptr (auto\_ptr\_ref< element\_type > __ref) throw ()`
- `~auto\_ptr ()`
- `element\_type * get () const throw ()`
- `template<typename _Tp1 >  
operator auto\_ptr<_Tp1 > () throw ()`
- `template<typename _Tp1 >  
operator auto\_ptr\_ref<_Tp1 > () throw ()`
- `element\_type & operator\* () const throw ()`
- `element\_type * operator-> () const throw ()`
- `auto\_ptr & operator= (auto\_ptr &__a) throw ()`
- `template<typename _Tp1 >  
auto\_ptr & operator= (auto\_ptr<_Tp1 > &__a) throw ()`
- `auto\_ptr & operator= (auto\_ptr\_ref< element\_type > __ref) throw ()`
- `element\_type * release () throw ()`
- `void reset (element\_type *__p=0) throw ()`

#### 4.275.1 Detailed Description

```
template<typename _Tp>
class std::auto_ptr< _Tp >
```

A simple smart pointer providing strict ownership semantics.

The Standard says:

An `auto_ptr` owns the object it holds a pointer to. Copying an `auto_ptr` copies the pointer and transfers ownership to the destination. If more than one `auto_ptr` owns the same object at the same time the behavior of the program is undefined.

The uses of `auto_ptr` include providing temporary exception-safety for dynamically allocated memory, passing ownership of dynamically allocated memory to a function, and returning dynamically allocated memory from a function. `auto_ptr` does not meet the CopyConstructible requirements for Standard Library `container` elements and thus instantiating a Standard Library container with an `auto_ptr` results in undefined behavior.

Quoted from [20.4.5]/3.

Good examples of what can and cannot be done with `auto_ptr` can be found in the libstdc++ testsuite.

`_GLIBCXX_RESOLVE_LIB_DEFECTS`

1. `auto_ptr<>` conversion issues These resolutions have all been incorporated.

Definition at line 89 of file `auto_ptr.h`.

#### 4.275.2 Member Typedef Documentation

##### 4.275.2.1 `element_type`

```
template<typename _Tp>
typedef _Tp std::auto_ptr< _Tp >::element_type
```

The pointed-to type.

Definition at line 96 of file `auto_ptr.h`.


#### 4.275.3 Constructor & Destructor Documentation

##### 4.275.3.1 `auto_ptr()` [1/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 element_type * __p = 0) throw () [inline], [explicit]
```

An `auto_ptr` is usually constructed from a raw pointer.

## Parameters

|                                                                                                       |                               |
|-------------------------------------------------------------------------------------------------------|-------------------------------|
| <br><code>__p</code> | A pointer (defaults to NULL). |
|-------------------------------------------------------------------------------------------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`.


Definition at line 105 of file `auto_ptr.h`.

## 4.275.3.2 auto\_ptr() [2/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr< _Tp > & __a) throw () [inline]
```

An `auto_ptr` can be constructed from another `auto_ptr`.

## Parameters

|                                                                                                       |                                                 |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <br><code>__a</code> | Another <code>auto_ptr</code> of the same type. |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|

This object now *owns* the object previously owned by `__a`, which has given up ownership.


Definition at line 114 of file `auto_ptr.h`.

## 4.275.3.3 auto\_ptr() [3/4]

```
template<typename _Tp>
template<typename _Tp1 >
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr< _Tp1 > & __a) throw () [inline]
```

An `auto_ptr` can be constructed from another `auto_ptr`.

## Parameters

|                                                                                                         |                                                                |
|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| <br><code>__a</code> | Another <code>auto_ptr</code> of a different but related type. |
|---------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|

A pointer-to-`Tp1` must be convertible to a pointer-to-`Tp/element_type`.

This object now *owns* the object previously owned by `__a`, which has given up ownership.

Definition at line 127 of file `auto_ptr.h`.

#### 4.275.3.4 ~auto\_ptr()

```
template<typename _Tp>
std::auto_ptr< _Tp >::~~auto_ptr () [inline]
```

When the auto\_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., get () is NULL), then this has no effect.

The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if \_Tp::~~\_Tp() throws, but this is prohibited. [17.4.3.6]/2

Definition at line 172 of file auto\_ptr.h.

#### 4.275.3.5 auto\_ptr() [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
 auto_ptr_ref< element_type > __ref) throw () [inline]
```

Automatic conversions.

These operations are supposed to convert an auto\_ptr into and from an auto\_ptr\_ref automatically as needed. This would allow constructs such as

```
auto_ptr<Derived> func_returning_auto_ptr(...);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(...);
```

But it doesn't work, and won't be fixed. For further details see <http://cplusplus.github.io/LWG/G/lwg-closed.html#463>

Definition at line 266 of file auto\_ptr.h.

### 4.275.4 Member Function Documentation

#### 4.275.4.1 get()

```
template<typename _Tp>
element_type* std::auto_ptr< _Tp >::get (
 void) const throw () [inline]
```

Bypassing the smart pointer.

##### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

##### Note

This auto\_ptr still owns the memory.

Definition at line 213 of file auto\_ptr.h.



## 4.275.4.2 operator\*()

```
template<typename _Tp>
element_type& std::auto_ptr< _Tp >::operator* () const throw () [inline]
```

Smart pointer dereferencing.

If this auto\_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

Definition at line 183 of file auto\_ptr.h.

## 4.275.4.3 operator-&gt;()

```
template<typename _Tp>
element_type* std::auto_ptr< _Tp >::operator-> () const throw () [inline]
```

Smart pointer dereferencing.

This returns the pointer itself, which the language then will automatically cause to be dereferenced.


Definition at line 196 of file auto\_ptr.h.

## 4.275.4.4 operator=( ) [1/2]

```
template<typename _Tp>
auto_ptr& std::auto_ptr< _Tp >::operator= (
 auto_ptr< _Tp > & __a) throw () [inline]
```

auto\_ptr assignment operator.

## Parameters

|                                                                                            |                                    |
|--------------------------------------------------------------------------------------------|------------------------------------|
| <br>__a | Another auto_ptr of the same type. |
|--------------------------------------------------------------------------------------------|------------------------------------|

This object now *owns* the object previously owned by \_\_a, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 138 of file auto\_ptr.h.

References std::auto\_ptr< \_Tp >::reset().

#### 4.275.4.5 operator=() [2/2]

```
template<typename _Tp>
template<typename _Tp1 >
auto_ptr& std::auto_ptr< _Tp >::operator= (
 auto_ptr< _Tp1 > & __a) throw () [inline]
```

auto\_ptr assignment operator.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__a</code> | Another auto_ptr of a different but related type. |
|------------------|---------------------------------------------------|

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element\_type.

This object now *owns* the object previously owned by \_\_a, which has given up ownership. The object that this one *used* to own and track has been deleted.

Definition at line 156 of file auto\_ptr.h.

References std::auto\_ptr< \_Tp >::reset().

#### 4.275.4.6 release()

```
template<typename _Tp>
element_type* std::auto_ptr< _Tp >::release () throw () [inline]
```

Bypassing the smart pointer.

##### Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

##### Note

This auto\_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

Definition at line 227 of file auto\_ptr.h.

#### 4.275.4.7 reset()

```
template<typename _Tp>
void std::auto_ptr< _Tp >::reset (
 element_type * __p = 0) throw () [inline]
```

Forcibly deletes the managed object.

## Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__p</code> | A pointer (defaults to NULL). |
|------------------|-------------------------------|

This object now *owns* the object pointed to by `__p`. The previous object has been deleted.

Definition at line 242 of file `auto_ptr.h`.

Referenced by `std::auto_ptr< _Tp >::operator=()`.

The documentation for this class was generated from the following file:

- [auto\\_ptr.h](#)

4.276 `std::auto_ptr_ref< _Tp1 >` Struct Template Reference

## Public Member Functions

- `auto_ptr_ref ( _Tp1 * __p )`

## Public Attributes

- `_Tp1 * _M_ptr`

## 4.276.1 Detailed Description

```
template<typename _Tp1>
struct std::auto_ptr_ref< _Tp1 >
```

A wrapper class to provide `auto_ptr` with reference semantics. For example, an `auto_ptr` can be assigned (or constructed from) the result of a function which returns an `auto_ptr` by value.

All the `auto_ptr_ref` stuff should happen behind the scenes.

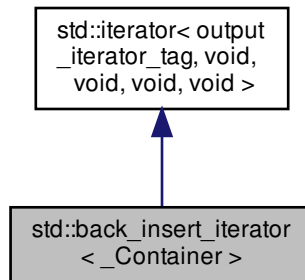
Definition at line 48 of file `auto_ptr.h`.

The documentation for this struct was generated from the following file:

- [auto\\_ptr.h](#)

## 4.277 `std::back_insert_iterator<_Container>` Class Template Reference

Inheritance diagram for `std::back_insert_iterator<_Container>`:



### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- constexpr `back_insert_iterator` (`_Container` &\_\_x)
- constexpr `back_insert_iterator` & `operator*` ()
- constexpr `back_insert_iterator` & `operator++` ()
- constexpr `back_insert_iterator` `operator++` (int)
- constexpr `back_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- constexpr `back_insert_iterator` & `operator=` (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container` \* **`container`**

#### 4.277.1 Detailed Description

```
template<typename _Container>
class std::back_insert_iterator<_Container>
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push\_back.

Tip: Using the back\_inserter function to create these iterators can save typing.

Definition at line 596 of file bits/stl\_iterator.h.

#### 4.277.2 Member Typedef Documentation

##### 4.277.2.1 container\_type

```
template<typename _Container>
typedef _Container std::back_insert_iterator<_Container>::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 604 of file bits/stl\_iterator.h.

##### 4.277.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file stl\_iterator\_base\_types.h.

##### 4.277.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.277.2.4 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file `stl_iterator_base_types.h`.

#### 4.277.2.5 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file `stl_iterator_base_types.h`.

#### 4.277.2.6 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

### 4.277.3 Constructor & Destructor Documentation

#### 4.277.3.1 back\_insert\_iterator()

```
template<typename _Container >
constexpr std::back_insert_iterator< _Container >::back_insert_iterator (
 _Container & __x) [inline], [explicit]
```

The only way to create this iterator is with a container.

Definition at line 613 of file `bits/stl_iterator.h`.

### 4.277.4 Member Function Documentation

#### 4.277.4.1 operator\*()

```
template<typename _Container >
constexpr back_insert_iterator& std::back_insert_iterator<_Container>::operator* () [inline]
```

Simply returns \*this.

Definition at line 655 of file bits/stl\_iterator.h.

#### 4.277.4.2 operator++() [1/2]

```
template<typename _Container >
constexpr back_insert_iterator& std::back_insert_iterator<_Container>::operator++ () [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 661 of file bits/stl\_iterator.h.

#### 4.277.4.3 operator++() [2/2]

```
template<typename _Container >
constexpr back_insert_iterator std::back_insert_iterator<_Container>::operator++ (
 int) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 667 of file bits/stl\_iterator.h.

#### 4.277.4.4 operator=()

```
template<typename _Container >
constexpr back_insert_iterator& std::back_insert_iterator<_Container>::operator= (
 const typename _Container::value_type & __value) [inline]
```

##### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.

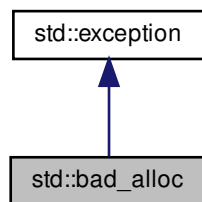
Definition at line 637 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 4.278 `std::bad_alloc` Class Reference

Inheritance diagram for `std::bad_alloc`:



### Public Member Functions

- **`bad_alloc`** (const [bad\\_alloc](#) &)=default
- [bad\\_alloc](#) & **`operator=`** (const [bad\\_alloc](#) &)=default
- virtual const char \* [what](#) () const throw ()

### 4.278.1 Detailed Description

Exception possibly thrown by `new`.

`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

Definition at line 55 of file `new`.

### 4.278.2 Member Function Documentation



4.278.2.1 `what()`

```
virtual const char* std::bad_alloc::what () const throw () [virtual]
```

Returns a C-style character string describing the general cause of the current error.

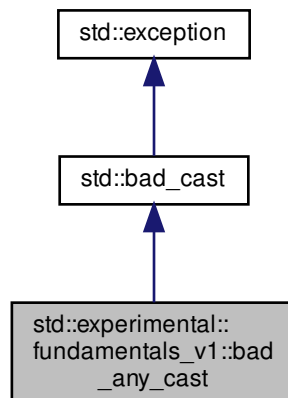
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [new](#)

4.279 `std::experimental::fundamentals_v1::bad_any_cast` Class Reference

Inheritance diagram for `std::experimental::fundamentals_v1::bad_any_cast`:



## Public Member Functions

- virtual const char \* [what](#) () const noexcept

## 4.279.1 Detailed Description

Exception class thrown by a failed `any_cast`.

Definition at line 67 of file `experimental/any`.

## 4.279.2 Member Function Documentation

### 4.279.2.1 what()

```
virtual const char* std::experimental::fundamentals_v1::bad_any_cast::what () const [inline],
[virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::bad\\_cast](#).

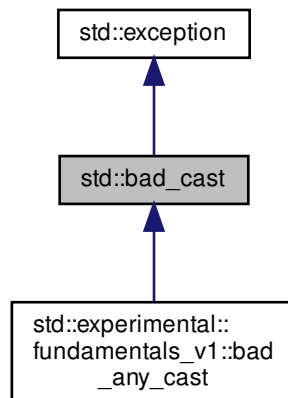
Definition at line 70 of file experimental/any.

The documentation for this class was generated from the following file:

- [experimental/any](#)

## 4.280 std::bad\_cast Class Reference

Inheritance diagram for std::bad\_cast:



### Public Member Functions

- virtual const char \* [what](#) ( ) const noexcept

## 4.280.1 Detailed Description

Thrown during incorrect typecasting.

If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

Definition at line 190 of file `typeinfo`.

## 4.280.2 Member Function Documentation

4.280.2.1 `what()`

```
virtual const char* std::bad_cast::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

Reimplemented from [std::exception](#).

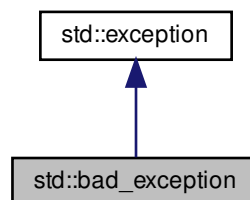
Reimplemented in [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

4.281 `std::bad_exception` Class Reference

Inheritance diagram for `std::bad_exception`:



## Public Member Functions

- virtual const char \* [what](#) ( ) const noexcept

#### 4.281.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

Definition at line 50 of file exception.

#### 4.281.2 Member Function Documentation

##### 4.281.2.1 what()

```
virtual const char* std::bad_exception::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

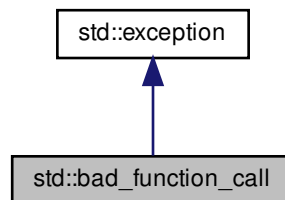
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [exception](#)

#### 4.282 std::bad\_function\_call Class Reference

Inheritance diagram for std::bad\_function\_call:



#### Public Member Functions

- const char \* [what](#) () const noexcept

#### 4.282.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

Definition at line 56 of file std\_function.h.

#### 4.282.2 Member Function Documentation

##### 4.282.2.1 what()

```
const char* std::bad_function_call::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

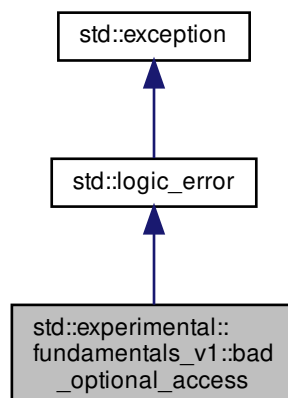
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

#### 4.283 std::experimental::fundamentals\_v1::bad\_optional\_access Class Reference

Inheritance diagram for std::experimental::fundamentals\_v1::bad\_optional\_access:



## Public Member Functions

- **bad\_optional\_access** (const char \*\_\_arg)
- virtual const char \* [what](#) () const noexcept

### 4.283.1 Detailed Description

Exception class thrown when a disengaged optional object is dereferenced.

Definition at line 104 of file experimental/optional.

### 4.283.2 Member Function Documentation

#### 4.283.2.1 what()

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

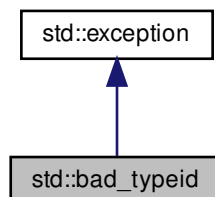
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [experimental/optional](#)

## 4.284 std::bad\_typeid Class Reference

Inheritance diagram for std::bad\_typeid:



### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 4.284.1 Detailed Description

Thrown when a NULL pointer in a typeid expression is used.

Definition at line 207 of file typeinfo.

#### 4.284.2 Member Function Documentation

##### 4.284.2.1 what()

```
virtual const char* std::bad_typeid::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

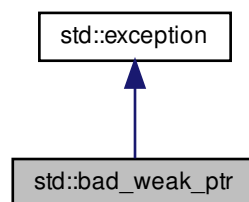
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 4.285 std::bad\_weak\_ptr Class Reference

Inheritance diagram for std::bad\_weak\_ptr:



### Public Member Functions

- virtual char const \* [what](#) () const noexcept

#### 4.285.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

Definition at line 76 of file `shared_ptr_base.h`.

#### 4.285.2 Member Function Documentation

##### 4.285.2.1 `what()`

```
virtual char const* std::bad_weak_ptr::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.

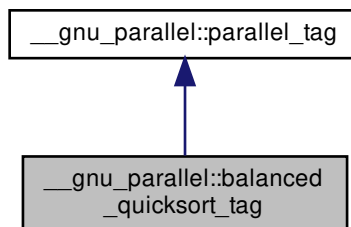
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [shared\\_ptr\\_base.h](#)

#### 4.286 `__gnu_parallel::balanced_quicksort_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::balanced_quicksort_tag`:



#### Public Member Functions

- **`balanced_quicksort_tag`** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) \_\_num\_threads)



#### 4.286.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

Definition at line 164 of file tags.h.

#### 4.286.2 Member Function Documentation

##### 4.286.2.1 `__get_num_threads()`

```
__ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 4.286.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
 __ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

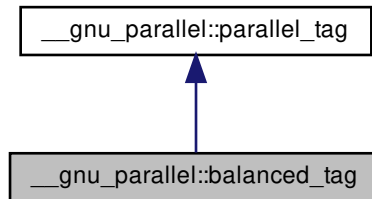
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.287 \_\_gnu\_parallel::balanced\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::balanced\_tag:



### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

#### 4.287.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

Definition at line 88 of file tags.h.

#### 4.287.2 Member Function Documentation

##### 4.287.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

#### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 4.287.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.288 `std::tr2::bases< _Tp >` Struct Template Reference

## Public Types

- typedef [\\_\\_reflection\\_typelist](#)< `__bases(_Tp)...` > **type**

## 4.288.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::bases< _Tp >
```

Sequence abstraction metafunctions for manipulating a typelist.

Enumerate all the base classes of a class. Form of a typelist.

Definition at line 88 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

4.289 `__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_TI, _Alloc >` Class Template Reference

Inherits type< `Key, Mapped, _Alloc, Tag, Policy_TI` >.

## Public Types

- typedef `Node_Update` **node\_update**

## Protected Member Functions

- **basic\_branch** (const [basic\\_branch](#) &other)
- `template<typename T0 >`  
**basic\_branch** (T0 t0)
- `template<typename T0 , typename T1 >`  
**basic\_branch** (T0 t0, T1 t1)
- `template<typename T0 , typename T1 , typename T2 >`  
**basic\_branch** (T0 t0, T1 t1, T2 t2)
- `template<typename T0 , typename T1 , typename T2 , typename T3 >`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 >`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 >`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0 , typename T1 , typename T2 , typename T3 , typename T4 , typename T5 , typename T6 >`  
**basic\_branch** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

### 4.289.1 Detailed Description

```
template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
```

A branched, tree-like (tree, trie) container abstraction.

#### Template Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                           |
| <i>Mapped</i>      | Map type.                                                           |
| <i>Tag</i>         | Instantiating data structure type, see <code>container_tag</code> . |
| <i>Node_Update</i> | Updates nodes, restores invariants.                                 |
| <i>Policy_Tl</i>   | Policy typelist.                                                    |
| <i>_Alloc</i>      | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `tree_tag`, `trie_tag`, and their descendants.

Base choices are: `detail::ov_tree_map`, `detail::rb_tree_map`, `detail::splay_tree_map`, and `detail::pat_trie_map`.

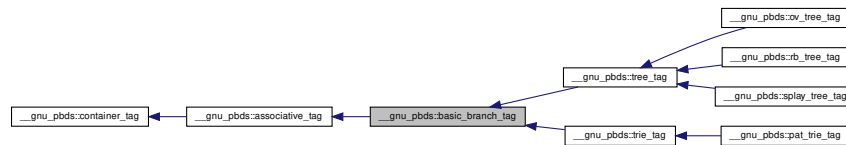
Definition at line 555 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.290 `__gnu_pbds::basic_branch_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_branch_tag`:



## 4.290.1 Detailed Description

Basic branch structure.

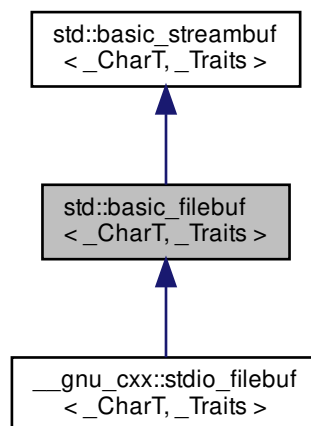
Definition at line 147 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.291 `std::basic_filebuf<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `std::basic_filebuf<_CharT, _Traits>`:



## Public Types

- typedef `codecvt`< char\_type, char, \_\_state\_type > **\_\_codecvt\_type**
- typedef \_\_basic\_file< char > **\_\_file\_type**
- typedef `basic_filebuf`< char\_type, traits\_type > **\_\_filebuf\_type**
- typedef traits\_type::state\_type **\_\_state\_type**
- typedef `basic_streambuf`< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

## Public Member Functions

- `basic_filebuf` ()
  - **basic\_filebuf** (const `basic_filebuf` &)=delete
  - **basic\_filebuf** (`basic_filebuf` &&)
  - virtual `~basic_filebuf` ()
  - `__filebuf_type` \* `close` ()
  - `locale` `getloc` () const
  - `streamsize` `in_avail` ()
  - bool `is_open` () const throw ()
  - `__filebuf_type` \* `open` (const char \*\_\_s, `ios_base::openmode` \_\_mode)
  - `__filebuf_type` \* `open` (const `std::string` &\_\_s, `ios_base::openmode` \_\_mode)
  - `basic_filebuf` & **operator=** (const `basic_filebuf` &)=delete
  - `basic_filebuf` & **operator=** (`basic_filebuf` &&)
  - `locale` `pubimbue` (const `locale` &\_\_loc)
  - int\_type `sputc` ()
  - int\_type `sgetc` ()
  - `streamsize` `sgetn` (char\_type \*\_\_s, `streamsize` \_\_n)
  - int\_type `snextc` ()
  - int\_type `sputbackc` (char\_type \_\_c)
  - int\_type `sputc` (char\_type \_\_c)
  - `streamsize` `sputn` (const char\_type \*\_\_s, `streamsize` \_\_n)
  - int\_type `sungetc` ()
  - void **swap** (`basic_filebuf` &)
- 
- `basic_streambuf` \* `pubsetbuf` (char\_type \*\_\_s, `streamsize` \_\_n)
  - pos\_type `pubseekoff` (off\_type \_\_off, `ios_base::seekdir` \_\_way, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
  - pos\_type `pubseekpos` (pos\_type \_\_sp, `ios_base::openmode` \_\_mode=`ios_base::in`|`ios_base::out`)
  - int `pubsync` ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
  - void **\_\_safe\_pbump** (streamsize \_\_n)
  - void **\_M\_allocate\_internal\_buffer** ()
  - bool **\_M\_convert\_to\_external** (char\_type \*, streamsize)
  - void **\_M\_create\_pback** ()
  - void **\_M\_destroy\_internal\_buffer** () throw ()
  - void **\_M\_destroy\_pback** () throw ()
  - int **\_M\_get\_ext\_pos** (\_\_state\_type &\_\_state)
  - pos\_type **\_M\_seek** (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
  - void **\_M\_set\_buffer** (streamsize \_\_off)
  - bool **\_M\_terminate\_output** ()
  - void **gbump** (int \_\_n)
  - virtual void **imbue** (const locale &\_\_loc)
  - virtual int\_type **overflow** (int\_type \_\_c=\_Traits::eof())
  - virtual int\_type **pbackfail** (int\_type \_\_c=\_Traits::eof())
  - void **pbump** (int \_\_n)
  - virtual pos\_type **seekoff** (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type **seekpos** (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual **\_\_streambuf\_type** \* **setbuf** (char\_type \* \_\_s, streamsize \_\_n)
  - void **setg** (char\_type \* \_\_gbeg, char\_type \* \_\_gnext, char\_type \* \_\_gend)
  - void **setp** (char\_type \* \_\_pbeg, char\_type \* \_\_pend)
  - virtual **streamsize showmanyc** ()
  - void **swap** (basic\_streambuf &\_\_sb)
  - virtual int **sync** ()
  - virtual int\_type **uflow** ()
  - virtual int\_type **underflow** ()
  - virtual **streamsize xsgetn** (char\_type \* \_\_s, streamsize \_\_n)
  - virtual **streamsize xsputn** (const char\_type \* \_\_s, streamsize \_\_n)
- 
- char\_type \* **eback** () const
  - char\_type \* **gptr** () const
  - char\_type \* **egptr** () const
- 
- char\_type \* **pbase** () const
  - char\_type \* **pptr** () const
  - char\_type \* **epptr** () const

## Protected Attributes

- `char_type * _M_buf`
  - `bool _M_buf_allocated`
  - `locale _M_buf_locale`
  - `size_t _M_buf_size`
  - `const __codecvt_type * _M_codecvt`
  - `char * _M_ext_buf`
  - `streamsize _M_ext_buf_size`
  - `char * _M_ext_end`
  - `const char * _M_ext_next`
  - `__file_type _M_file`
  - `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `__c_lock _M_lock`
  - `ios_base::openmode _M_mode`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - `bool _M_reading`
  - `__state_type _M_state_beg`
  - `__state_type _M_state_cur`
  - `__state_type _M_state_last`
  - `bool _M_writing`
- 
- `char_type _M_pback`
  - `char_type * _M_pback_cur_save`
  - `char_type * _M_pback_end_save`
  - `bool _M_pback_init`

## Friends

- class **ios\_base**

## 4.291.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >
```

The actual work of input and output (for files).

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |



This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.

Requirements on `traits_type`, specific to this class:

- `traits_type::pos_type` must be `fpos<traits_type::state_type>`
- `traits_type::off_type` must be `streamoff`
- `traits_type::state_type` must be Assignable and DefaultConstructible,
- `traits_type::state_type()` must be the initial state for `codecvt`.

Definition at line 80 of file `fstream`.

#### 4.291.2 Constructor & Destructor Documentation

##### 4.291.2.1 `basic_filebuf()`

```
template<typename _CharT, typename _Traits >
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ()
```

Does not open any files.

The default constructor initializes the parent class using its own default ctor.

Definition at line 81 of file `fstream.tcc`.

##### 4.291.2.2 `~basic_filebuf()`

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~~basic_filebuf () [inline], [virtual]
```

The destructor closes the file first.

Definition at line 246 of file `fstream`.

#### 4.291.3 Member Function Documentation

#### 4.291.3.1 `_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback () [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

#### 4.291.3.2 `_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw () [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

#### 4.291.3.3 `_M_set_buffer()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
 streamsize __off) [inline], [protected]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `epptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 459 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 4.291.3.4 `close()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()
```

Closes the currently associated file.

##### Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 250 of file `fstream.tcc`.

#### 4.291.3.5 `eback()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 4.291.3.6 `egptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

#### 4.291.3.7 `epptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 542 of file `streambuf`.

#### 4.291.3.8 `gbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**4.291.3.9 getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**4.291.3.10 gptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**4.291.3.11 imbue()**

```
template<typename _CharT , typename _Traits >
void std::basic_filebuf< _CharT, _Traits >::imbue (
 const locale & __loc) [protected], [virtual]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 1030 of file fstream.tcc.

## 4.291.3.12 in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

## 4.291.3.13 is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw () [inline]
```

Returns true if the external file is open.

Definition at line 265 of file fstream.

## 4.291.3.14 open() [1/2]

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
 const char * __s,
 ios_base::openmode __mode)
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

Definition at line 180 of file `fstream.tcc`.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

4.291.3.15 `open()` [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, `NULL` on failure

Definition at line 331 of file `fstream`.

4.291.3.16 `overflow()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
 int_type __c = _Traits::eof()) [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

## Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 542 of file `fstream.tcc`.

4.291.3.17 `pbackfail()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_c</code> | The character to be inserted back into the sequence. |
|-----------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 483 of file `fstream.tcc`.

**4.291.3.18 pbase()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

**4.291.3.19 pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.



## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

## 4.291.3.20 pptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

## 4.291.3.21 pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

#### 4.291.3.22 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

##### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 4.291.3.23 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

##### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

#### 4.291.3.24 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 4.291.3.25 pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 4.291.3.26 sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.291.3.27 seekoff()

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 833 of file `fstream.tcc`.

**4.291.3.28 seekpos()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 893 of file `fstream.tcc`.

**4.291.3.29 setbuf()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 804 of file `fstream.tcc`.

## 4.291.3.30 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

## 4.291.3.31 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

## Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

**4.291.3.32 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

**4.291.3.33 sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**4.291.3.34 showmanyc()**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc () [protected], [virtual]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 298 of file `fstream.tcc`.

**4.291.3.35 snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**4.291.3.36 sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.291.3.37 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**4.291.3.38 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.



## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

4.291.3.39 `sungetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

## Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

4.291.3.40 `sync()`

```
template<typename _CharT , typename _Traits >
int std::basic_filebuf< _CharT, _Traits >::sync () [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

## Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

## Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 1013 of file `fstream.tcc`.

#### 4.291.3.41 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

#### 4.291.3.42 underflow()

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 324 of file `fstream.tcc`.

#### 4.291.3.43 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 670 of file `fstream.tcc`.

## 4.291.3.44 xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 756 of file `fstream.tcc`.

#### 4.291.4 Member Data Documentation

##### 4.291.4.1 `_M_buf`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf [protected]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file `fstream`.

##### 4.291.4.2 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file `streambuf`.

##### 4.291.4.3 `_M_buf_size`

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file `fstream`.

##### 4.291.4.4 `_M_ext_buf`

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file `fstream`.

#### 4.291.4.5 \_M\_ext\_buf\_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected]
```

Size of buffer held by \_M\_ext\_buf.

Definition at line 183 of file fstream.

#### 4.291.4.6 \_M\_ext\_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected]
```

Pointers into the buffer held by \_M\_ext\_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, \_M\_ext\_next corresponds to egptr().

Definition at line 190 of file fstream.

#### 4.291.4.7 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 4.291.4.8 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

#### 4.291.4.9 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 4.291.4.10 `_M_mode`

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file fstream.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 4.291.4.11 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.291.4.12 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.291.4.13 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

#### 4.291.4.14 \_M\_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file fstream.

#### 4.291.4.15 \_M\_pback\_cur\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 4.291.4.16 \_M\_pback\_end\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

#### 4.291.4.17 `_M_pback_init`

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 167 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 4.291.4.18 `_M_reading`

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected]
```

`_M_reading == false && _M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true && _M_writing == true` is unused.

Definition at line 155 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

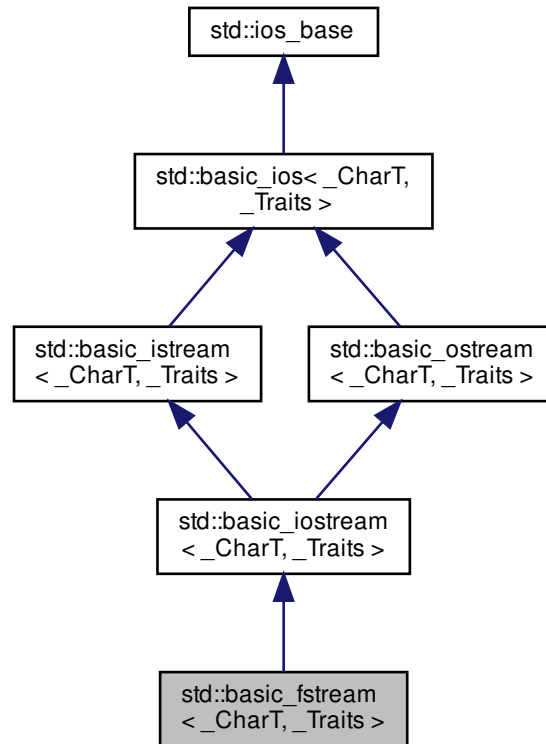
The documentation for this class was generated from the following files:

- [fstream](#)
- [fstream.tcc](#)



## 4.292 std::basic\_fstream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_fstream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< char_type, traits_type > __ios_type`
- typedef `basic_iostream< char_type, traits_type > __iostream_type`
- typedef `basic_istream< _CharT, _Traits > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< _CharT, _Traits > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `int io_state _GLIBCXX_DEPRECATED_SUGGEST("std::iostate")`
- typedef `int open_mode _GLIBCXX_DEPRECATED_SUGGEST("std::openmode")`

- typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::seekdir")
  - typedef [std::streampos](#) streampos **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streampos")
  - typedef [std::streamoff](#) streamoff **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streamoff")
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags [fmtflags](#)
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_istate [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_fstream](#) ()
- [basic\\_fstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- [basic\\_fstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- **basic\_fstream** (const [basic\\_fstream](#) &)=delete
- **basic\_fstream** ([basic\\_fstream](#) && \_\_rhs)
- ~[basic\\_fstream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT & \_\_v)
- const [locale](#) & **\_M\_getloc** () const
- template<typename \_ValueT >  
[basic\\_ostream](#)< \_CharT, \_Traits > & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< char > & [getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)

- `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __default) const`
  - `void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `basic_fstream & operator= (const basic_fstream &)=delete`
  - `basic_fstream & operator= (basic_fstream &&__rhs)`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `__filebuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void swap (basic_fstream &__rhs)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`

- `__istream_type & operator>> (ios_base &(*__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`
  - `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type & __sb, char_type __delim)`
- `__istream_type & get (__streambuf_type & __sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`

- [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
[\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
[\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_iostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.292.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_fstream< _CharT, _Traits >
```

Controlling input and output for files.

### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from and writing to named files, using the inherited functions from `std::basic_iostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 1016 of file `fstream`.

## 4.292.2 Member Typedef Documentation

### 4.292.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 4.292.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.



#### 4.292.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.292.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 416 of file `ios_base.h`.

#### 4.292.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.292.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.292.3 Member Enumeration Documentation

#### 4.292.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

## 4.292.4 Constructor &amp; Destructor Documentation

## 4.292.4.1 basic\_fstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 1043 of file `fstream`.

## 4.292.4.2 basic\_fstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

Definition at line 1053 of file `fstream`.

## 4.292.4.3 basic\_fstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_fstream< _CharT, _Traits >::basic_fstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Create an input/output file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

Definition at line 1083 of file fstream.

#### 4.292.4.4 ~basic\_fstream()

```
template<typename _CharT, typename _Traits >
std::basic_fstream< _CharT, _Traits >::~~basic_fstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 1118 of file fstream.

### 4.292.5 Member Function Documentation

#### 4.292.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 804 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

#### 4.292.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>_↵<br/>_s</code> | The array to insert.                    |
| <code>_↵<br/>_n</code> | Maximum number of characters to insert. |

Definition at line 317 of file ostream.

**4.292.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.292.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 4.292.5.5 close()

```
template<typename _CharT, typename _Traits >
void std::basic_fstream< _CharT, _Traits >::close () [inline]
```

Close the file.

Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1253 of file `fstream`.

#### 4.292.5.6 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

##### Returns

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

#### 4.292.5.7 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the `eofbit` is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

## 4.292.5.8 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 4.292.5.9 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios\_base::failure is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file basic\_ios.h.

#### 4.292.5.10 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::operator bool(), and std::basic\_ios< char, \_Traits >::operator!().

#### 4.292.5.11 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

##### Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::basic\_ios< char, \_Traits >::fill(), and std::operator<<().

#### 4.292.5.12 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

##### Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|



**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

**4.292.5.13 flags()** [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**4.292.5.14 flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

**4.292.5.15 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

**4.292.5.16 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

**4.292.5.17 get()** [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets `failbit` and returns `traits::eof()`.

Definition at line 244 of file `istream.tcc`.

**4.292.5.18 get()** [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

## Returns

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

## 4.292.5.19 get() [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

**4.292.5.20 get()** [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

`*this`

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**4.292.5.21 get()** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**`*this`

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file `istream.tcc`.

**4.292.5.22 `get()`** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**`*this`

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file `istream`.

**4.292.5.23 `getline()`** [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**4.292.5.24 `getline()`** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

#### 4.292.5.25 `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

#### 4.292.5.26 `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

#### 4.292.5.27 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.292.5.28 `ignore()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**4.292.5.29 ignore()** [2/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.



## 4.292.5.30 ignore() [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

## 4.292.5.31 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

## 4.292.5.32 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

#### 4.292.5.33 `is_open()`

```
template<typename _CharT , typename _Traits >
bool std::basic_fstream< _CharT, _Traits >::is_open () [inline]
```

Wrapper to test for an open file.

##### Returns

`rdbuf() -> is_open()`

Definition at line 1159 of file `fstream`.

#### 4.292.5.34 `yword()`

```
long& std::ios_base::yword (
 int __ix) [inline], [inherited]
```

Access to integer array.

##### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

##### Returns

A reference to an integer associated with the index.

The `yword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

## 4.292.5.35 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

## Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

## Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

## 4.292.5.36 open() [1/2]

```
template<typename _CharT , typename _Traits >
void std::basic_fstream< _CharT, _Traits >::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1177 of file `fstream`.

**4.292.5.37** `open()` [2/2]

```
template<typename _CharT, typename _Traits >
void std::basic_fstream< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s, __mode)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 1218 of file `fstream`.

**4.292.5.38** `operator bool()`

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**4.292.5.39** `operator!()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

## 4.292.5.40 operator&lt;&lt;() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

## 4.292.5.41 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

## 4.292.5.42 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.


Definition at line 127 of file `ostream`.

## 4.292.5.43 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                   |                                      |
|-----------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                  |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**4.292.5.44 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                    |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.292.5.45 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 4.292.5.46 operator&lt;&lt;() [7/17]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 4.292.5.47 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**4.292.5.48 operator<<() [9/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**4.292.5.49 operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.



## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↔$ | A variable of builtin integral type. |
| $\_n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 4.292.5.50 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_↔$ | A variable of builtin integral type. |
| $\_n$ |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 4.292.5.51 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**4.292.5.52 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**4.292.5.53 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 4.292.5.54 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 4.292.5.55 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.292.5.56 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

## 4.292.5.57 operator&gt;&gt;() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

## 4.292.5.58 operator&gt;&gt;() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

## 4.292.5.59 operator&gt;&gt;() [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

## 4.292.5.60 operator&gt;&gt;() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| $\leftrightarrow$<br><code>_n</code> | A variable of builtin integral type. |
|--------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

**4.292.5.61 `operator>>()` [5/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| $\leftrightarrow$<br><code>_n</code> | A variable of builtin integral type. |
|--------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**4.292.5.62 `operator>>()` [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

## 4.292.5.63 operator&gt;&gt;() [7/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

## 4.292.5.64 operator&gt;&gt;() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                 |                                      |
|---------------------------------|--------------------------------------|
| $\leftarrow$<br><code>_n</code> | A variable of builtin integral type. |
|---------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.292.5.65 `operator>>()` [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                 |                                      |
|---------------------------------|--------------------------------------|
| $\leftarrow$<br><code>_n</code> | A variable of builtin integral type. |
|---------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.292.5.66 `operator>>()` [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.



## Parameters

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| <code>__n</code>  |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

## 4.292.5.67 operator&gt;&gt;() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                   |                                      |
|-------------------|--------------------------------------|
| $\leftrightarrow$ | A variable of builtin integral type. |
| <code>__n</code>  |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

## 4.292.5.68 operator&gt;&gt;() [12/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**4.292.5.69 operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**4.292.5.70 operator>>() [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

## 4.292.5.71 operator&gt;&gt;() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

## 4.292.5.72 operator&gt;&gt;() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.292.5.73 operator>>()** [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

## 4.292.5.74 peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

## Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

## 4.292.5.75 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

## 4.292.5.76 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().


Definition at line 728 of file ios\_base.h.

**4.292.5.77 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                                                                                   |                          |
|-----------------------------------------------------------------------------------|--------------------------|
|  | The character to insert. |
| <code>__c</code>                                                                  |                          |

**Returns**

\*this

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.


Definition at line 149 of file ostream.tcc.

**4.292.5.78 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                                                                                     |                                                   |
|-------------------------------------------------------------------------------------|---------------------------------------------------|
|  | The character to push back into the input stream. |
| <code>__c</code>                                                                    |                                                   |

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**4.292.5.79 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file `ios_base.h`.

**4.292.5.80 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.292.5.81 `rdbuf()`** [2/2]

```
template<typename _CharT, typename _Traits >
__filebuf_type* std::basic_fstream< _CharT, _Traits >::rdbuf () const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 1151 of file `fstream`.

**4.292.5.82 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.



## 4.292.5.83 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

## 4.292.5.84 readsome()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file `istream.tcc`.

**4.292.5.85 register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.292.5.86 seekg()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file istream.tcc.

**4.292.5.87 seekg()** [2/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file istream.tcc.

**4.292.5.88 seekp()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file `ostream.tcc`.

**4.292.5.89 seekp()** [2/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

**4.292.5.90 setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

**4.292.5.91 setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file ios\_base.h.

**4.292.5.92 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 4.292.5.93 `sync()`

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

##### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

#### 4.292.5.94 `sync_with_stdio()`

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**4.292.5.95 tellg()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, in).

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount(). At variance with putback, unget and seekg, eofbit is not cleared first.

Definition at line 825 of file istream.tcc.

**4.292.5.96 tellp()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If fail() is not false, returns pos\_type(-1) to indicate failure. Otherwise returns rdbuf()->pubseekoff(0, cur, out).

Definition at line 237 of file ostream.tcc.

**4.292.5.97** `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**4.292.5.98** `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.292.5.99** `unget()`

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
 void) [inherited]
```

Unextracting the previous character.



**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**4.292.5.100 unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.292.5.101 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.292.5.102 width() [1/2]**

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::num_put<_CharT, _OutIt>::do_put()`.

**4.292.5.103 width() [2/2]**

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file ios\_base.h.

#### 4.292.5.104 write()

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

##### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 4.292.5.105 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 4.292.6 Member Data Documentation

### 4.292.6.1 `_M_gcount`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, and `std::basic_istream< char >::unget()`.

### 4.292.6.2 `adjustfield`

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### 4.292.6.3 `app`

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

### 4.292.6.4 `ate`

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.292.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 4.292.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 399 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), std::oct(), and std::basic\_ostream< char >::operator<<().

#### 4.292.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.292.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.292.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_←put()`, and `std::noboolalpha()`.

#### 4.292.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 4.292.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.292.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 4.292.6.13 eofbit

```
const ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_istream< char >::unget()`, and `std::ws()`.

#### 4.292.6.14 failbit

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry()`.

#### 4.292.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::hexfloat()`.

#### 4.292.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.292.6.17 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

#### 4.292.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.292.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.



#### 4.292.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.292.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 4.292.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.292.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

**4.292.6.24 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

**4.292.6.25 scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

**4.292.6.26 showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::noshowbase(), and std::showbase().

**4.292.6.27 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.292.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.292.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.292.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.292.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

## 4.292.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [fstream](#)

4.293 `__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >` Class Template Reference

Inherits `type< Key, Mapped, _Alloc, Tag, __gnu_cxx::typelist::append< __gnu_cxx::typelist::create4< Hash_Fn, Eq_Fn, Resize_Policy, detail::integral_constant< int, Store_Hash > >::type, Policy_Tl >::type >`.

## Protected Member Functions

- **basic\_hash\_table** (const [basic\\_hash\\_table](#) &other)
- `template<typename T0 >`  
**basic\_hash\_table** (T0 t0)
- `template<typename T0, typename T1 >`  
**basic\_hash\_table** (T0 t0, T1 t1)
- `template<typename T0, typename T1, typename T2 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2)
- `template<typename T0, typename T1, typename T2, typename T3 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- `template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8 >`  
**basic\_hash\_table** (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

## 4.293.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, type-
name Tag, typename Policy_Tl, typename _Alloc>
```

```
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
```

A hashed container abstraction.

## Template Parameters

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                            |
| <i>Mapped</i>        | Map type.                                                            |
| <i>Hash_Fn</i>       | Hashing functor.                                                     |
| <i>Eq_Fn</i>         | Equal functor.                                                       |
| <i>Resize_Policy</i> | Resizes hash.                                                        |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. |
| <i>Tag</i>           | Instantiating data structure type, see <code>container_tag</code> .  |
| <i>Policy_TL</i>     | Policy typelist.                                                     |
| <i>_Alloc</i>        | Allocator type.                                                      |

Base is dispatched at compile time via `Tag`, from the following choices: `cc_hash_tag`, `gp_hash_tag`, and descendants of `basic_hash_tag`.

Base choices are: `detail::cc_ht_map`, `detail::gp_ht_map`

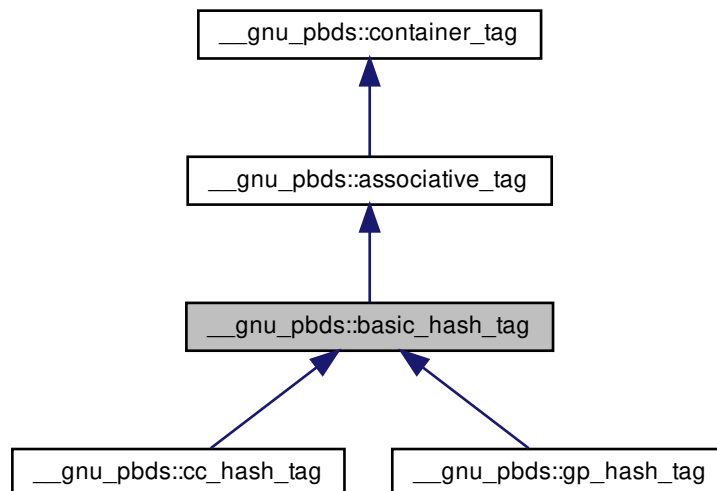
Definition at line 104 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.294 `__gnu_pbds::basic_hash_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::basic_hash_tag`:



#### 4.294.1 Detailed Description

Basic hash structure.

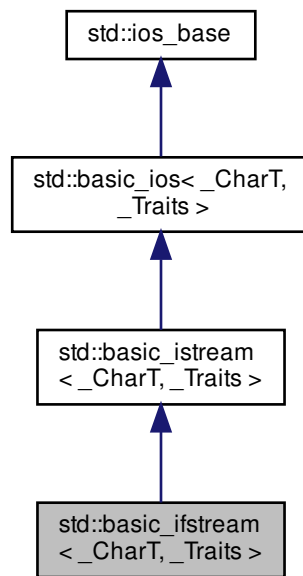
Definition at line 138 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.295 std::basic\_ifstream< \_CharT, \_Traits > Class Template Reference

Inheritance diagram for std::basic\_ifstream< \_CharT, \_Traits >:



#### Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_filebuf< char_type, traits_type > __filebuf_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `basic_istream< char_type, traits_type > __istream_type`
- typedef `num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `int io_state _GLIBCXX_DEPRECATED_SUGGEST("std::iostate")`

- typedef int open\_mode **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::openmode")
  - typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::seekdir")
  - typedef [std::streampos](#) **streampos** **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streampos")
  - typedef [std::streamoff](#) **streamoff** **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streamoff")
  - typedef \_CharT **char\_type**
  - enum [event](#) { **erase\_event**, **imbue\_event**, **copyfmt\_event** }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags **fmtflags**
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_istate **istate**
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode **openmode**
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

#### Public Member Functions

- [basic\\_ifstream](#) ()
- [basic\\_ifstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- [basic\\_ifstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_ifstream** (const [basic\\_ifstream](#) &)=delete
- **basic\_ifstream** ([basic\\_ifstream](#) &&\_\_rhs)
- [~basic\\_ifstream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([istate](#) \_\_state)
- bool **bad** () const
- void **clear** ([istate](#) \_\_state=[goodbit](#))
- void **close** ()
- [basic\\_ios](#) & **copyfmt** (const [basic\\_ios](#) &\_\_rhs)
- bool **eof** () const
- [istate exceptions](#) () const
- void **exceptions** ([istate](#) \_\_except)
- bool **fail** () const
- [char\\_type fill](#) () const
- [char\\_type fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags flags](#) () const
- [fmtflags flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize gcount](#) () const
- template<>  
[basic\\_istream](#)< char > & **getline** ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- template<>  
[basic\\_istream](#)< [wchar\\_t](#) > & **getline** ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)

- `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __default) const`
  - `void open (const char * __s, ios_base::openmode __mode=ios_base::in)`
  - `void open (const std::string & __s, ios_base::openmode __mode=ios_base::in)`
  - `basic_ifstream & operator= (const basic_ifstream &)=delete`
  - `basic_ifstream & operator= (basic_ifstream && __rhs)`
  - `__istream_type & operator>> (void *& __p)`
  - `__istream_type & operator>> (__streambuf_type * __sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `__filebuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void swap (basic_ifstream & __rhs)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(* __pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(* __pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(* __pf)(ios_base &))`



## Extractors

All the *operator>>* functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to `false`. This has several effects, concluding with the setting of a status flag; see the *sentry* documentation for more.

If the *sentry* status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
- `__istream_type & operator>> (short &__n)`
- `__istream_type & operator>> (unsigned short &__n)`
- `__istream_type & operator>> (int &__n)`
- `__istream_type & operator>> (unsigned int &__n)`
- `__istream_type & operator>> (long &__n)`
- `__istream_type & operator>> (unsigned long &__n)`
- `__istream_type & operator>> (long long &__n)`
- `__istream_type & operator>> (unsigned long long &__n)`

- `__istream_type & operator>> (float &__f)`
- `__istream_type & operator>> (double &__f)`
- `__istream_type & operator>> (long double &__f)`

## Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (*noskipws*) set to `true`. This has several effects, concluding with the setting of a status flag; see the *sentry* documentation for more.

If the *sentry* status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type * __s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type * __s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type * __s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`
- `__istream_type & read (char_type * __s, streamsize __n)`
- `streamsize readsome (char_type * __s, streamsize __n)`
- `__istream_type & putback (char_type __c)`
- `__istream_type & unget ()`
- `int sync ()`
- `pos_type tellg ()`

- [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
- [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, ios\_base::seekdir)

- [operator bool](#) () const
- bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.295.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ifstream< _CharT, _Traits >
```

Controlling input for files.

### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 492 of file `fstream`.

## 4.295.2 Member Typedef Documentation

### 4.295.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 4.295.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.295.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.295.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 416 of file `ios_base.h`.

#### 4.295.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.295.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.295.3 Member Enumeration Documentation

#### 4.295.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

## 4.295.4 Constructor &amp; Destructor Documentation

## 4.295.4.1 basic\_ifstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 518 of file `fstream`.

## 4.295.4.2 basic\_ifstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

## Parameters

|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.                |
| <code>__mode</code> | Open file in specified mode (see <code>std::ios_base</code> ). |

`ios_base::in` is automatically included in `__mode`.

Definition at line 529 of file `fstream`.

## 4.295.4.3 basic\_ifstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Create an input file stream.

**Parameters**

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | std::string specifying the filename.             |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::in` is automatically included in `__mode`.

Definition at line 562 of file `fstream`.

**4.295.4.4 ~basic\_ifstream()**

```
template<typename _CharT , typename _Traits >
std::basic_ifstream< _CharT, _Traits >::~~basic_ifstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 599 of file `fstream`.

**4.295.5 Member Function Documentation****4.295.5.1 \_M\_getloc()**

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.



## 4.295.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

## Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

## 4.295.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

## Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 4.295.5.4 close()

```
template<typename _CharT , typename _Traits >
void std::basic_ifstream< _CharT, _Traits >::close () [inline]
```

Close the file.

Calls std::basic\_filebuf::close(). If that function fails, failbit is set in the stream's error state.

Definition at line 730 of file fstream.

#### 4.295.5.5 copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 4.295.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 4.295.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::copyfmt().

## 4.295.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.295.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

**4.295.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.295.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.295.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**4.295.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

**4.295.5.14 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_ifstream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

**4.295.5.15** `get()` [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

**4.295.5.16** `get()` [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**4.295.5.17** `get()` [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file `istream.tcc`.

4.295.5.18 `get()` [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_ifstream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**4.295.5.19 get()** [5/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

**4.295.5.20 get()** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.



## Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

## Returns

`*this`

Returns `get(__sb, widen("\n"))`.

Definition at line 387 of file `istream`.

4.295.5.21 `getline()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

## Returns

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**4.295.5.22** `getline()` [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**4.295.5.23** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**4.295.5.24** `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, and `std::ws()`.

## 4.295.5.25 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around rdbuf.

Definition at line 180 of file basic\_ios.h.

Referenced by std::\_\_detail::operator>>(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

## 4.295.5.26 ignore() [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

## Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file istream.tcc.

**4.295.5.27 ignore()** [2/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 501 of file istream.tcc.

**4.295.5.28 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**4.295.5.29 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**4.295.5.30 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**4.295.5.31 is\_open()**

```
template<typename _CharT , typename _Traits >
bool std::basic_ifstream< _CharT, _Traits >::is_open () [inline]
```

Wrapper to test for an open file.

**Returns**

`rdbuf()->is_open()`

Definition at line 640 of file `fstream`.

**4.295.5.32 iword()**

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                        |                       |
|------------------------|-----------------------|
| <code>↵<br/>_ix</code> | Index into the array. |
|------------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

**4.295.5.33 narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

**Parameters**

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↵html>

Definition at line 430 of file `basic_ios.h`.

## 4.295.5.34 open() [1/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream<_CharT, _Traits>::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 658 of file `fstream`.

## 4.295.5.35 open() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream<_CharT, _Traits>::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::in) [inline]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 697 of file `fstream`.

## 4.295.5.36 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

**4.295.5.37 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**4.295.5.38 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 120 of file `istream`.

**4.295.5.39 operator>>() [2/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 124 of file `istream`.

**4.295.5.40 operator>>() [3/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `iomanip` header.

Definition at line 131 of file `istream`.

**4.295.5.41 operator>>() [4/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.



## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

## 4.295.5.42 operator&gt;&gt;() [5/17]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|                         |                                      |
|-------------------------|--------------------------------------|
| $\leftrightarrow$<br>_n | A variable of builtin integral type. |
|-------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

## 4.295.5.43 operator&gt;&gt;() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_ifstream<_CharT, _Traits>::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                 |                                      |
|---------------------------------|--------------------------------------|
| $\leftarrow$<br><code>_n</code> | A variable of builtin integral type. |
|---------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.295.5.44 `operator>>()` [7/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                 |                                      |
|---------------------------------|--------------------------------------|
| $\leftarrow$<br><code>_n</code> | A variable of builtin integral type. |
|---------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**4.295.5.45 `operator>>()` [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

## 4.295.5.46 operator&gt;&gt;() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

## 4.295.5.47 operator&gt;&gt;() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**4.295.5.48 operator>>() [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**4.295.5.49 operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| $\leftarrow$<br>_n | A variable of builtin integral type. |
|--------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

## 4.295.5.50 operator&gt;&gt;() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|                                                                          |                                            |
|--------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\leftarrow$<br>$\leftarrow$<br>$\leftarrow$<br><i>f</i> | A variable of builtin floating point type. |
|--------------------------------------------------------------------------|--------------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

## 4.295.5.51 operator&gt;&gt;() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**4.295.5.52 operator>>() [15/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**4.295.5.53 operator>>() [16/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_p</code> | A variable of pointer type. |
|-----------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

4.295.5.54 `operator>>()` [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

**4.295.5.55 peek()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

**Returns**

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

**4.295.5.56 precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**4.295.5.57 precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|



**Returns**

The previous value of precision().


Definition at line 728 of file ios\_base.h.

**4.295.558 putback()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                                                                                                              |                                                   |
|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| <br><b><code>__c</code></b> | The character to push back into the input stream. |
|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------|

**Returns**

\*this

If rdbuf() is not null, calls rdbuf()->sputbackc(c).

If rdbuf() is null or if sputbackc() fails, sets badbit in the error state.

**Note**

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.


Definition at line 719 of file istream.tcc.

**4.295.559 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                                                                                                                 |                       |
|-----------------------------------------------------------------------------------------------------------------|-----------------------|
| <br><b><code>__ix</code></b> | Index into the array. |
|-----------------------------------------------------------------------------------------------------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file `ios_base.h`.

**4.295.5.60 `rdbuf()`** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.295.5.61 `rdbuf()`** [2/2]

```
template<typename _CharT , typename _Traits >
__filebuf_type* std::basic_ifstream< _CharT, _Traits >::rdbuf () const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 632 of file `fstream`.

## 4.295.5.62 rdstate()

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, ↵  
\_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream< ↵  
char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 4.295.5.63 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

## Parameters

|          |                                        |
|----------|----------------------------------------|
| ↵<br>__s | A character array.                     |
| ↵<br>__n | Maximum number of characters to store. |

## Returns

\*this

If the stream state is good ( ) , extracts characters and stores them into \_\_s until one of the following happens:

- \_\_n characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to failbit|eofbit.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file istream.tcc.

4.295.5.64 `readsome()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file `istream.tcc`.

4.295.5.65 `register_callback()`

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 4.295.5.66 seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

##### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file istream.tcc.

#### 4.295.5.67 seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ifstream<_CharT, _Traits> & std::basic_ifstream<_CharT, _Traits>::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

##### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

#### 4.295.5.68 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

#### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

#### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 4.295.5.69 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

#### Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

**4.295.5.70 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**4.295.5.71 sync()**

```
template<typename _CharT , typename _Traits >
int std::basic_ifstream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf() ->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

#### 4.295.5.72 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 4.295.5.73 tellg()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

##### Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.



## 4.295.5.74 tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

## 4.295.5.75 tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

## 4.295.5.76 unget()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**4.295.5.77 unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.295.5.78 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.295.5.79 width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.295.5.80 width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file ios\_base.h.

#### 4.295.5.81 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 4.295.6 Member Data Documentation

#### 4.295.6.1 \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, and `std::basic_istream< char >::unget()`.

#### 4.295.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.295.6.3 `app`

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

#### 4.295.6.4 `ate`

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.295.6.5 `badbit`

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 4.295.6.6 `basefield`

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.295.6.7 `beg`

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file `ios_base.h`.

#### 4.295.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.295.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

#### 4.295.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 4.295.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by `std::dec()`.

## 4.295.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

## 4.295.6.13 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

## 4.295.6.14 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

## 4.295.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().

#### 4.295.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.295.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

#### 4.295.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.295.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.



#### 4.295.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.295.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 4.295.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.295.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

#### 4.295.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

#### 4.295.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

#### 4.295.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put(), std::noshowbase(), and std::showbase().

#### 4.295.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.295.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.295.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.295.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.295.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

#### 4.295.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file `ios_base.h`.

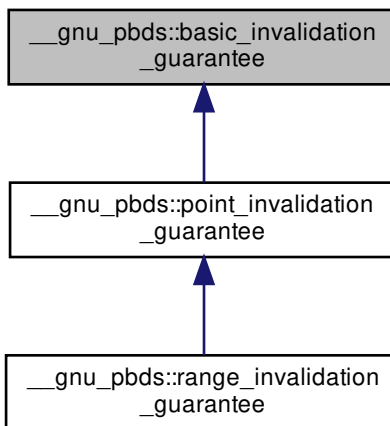
Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [fstream](#)

#### 4.296 \_\_gnu\_pbds::basic\_invalidation\_guarantee Struct Reference

Inheritance diagram for `__gnu_pbds::basic_invalidation_guarantee`:



##### 4.296.1 Detailed Description

Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.

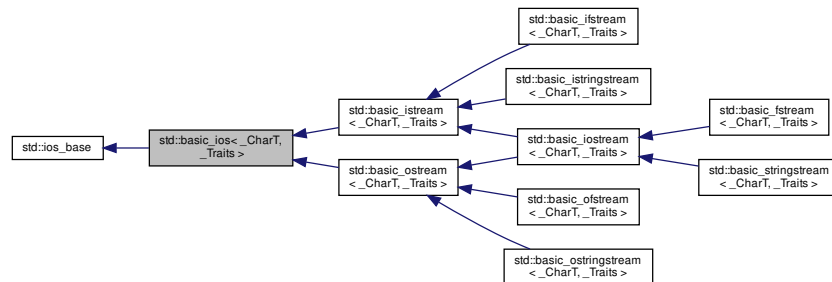
Definition at line 93 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.297 std::basic\_ios&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_ios< \_CharT, \_Traits >:



## Public Types

- typedef int io\_state **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::iostate")
- typedef int open\_mode **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::openmode")
- typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::seekdir")
- typedef [std::streampos](#) streampos **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streampos")
- typedef [std::streamoff](#) streamoff **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streamoff")
- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags [fmtflags](#)
- typedef \_ios\_Iostate [iostate](#)
- typedef \_ios\_Openmode [openmode](#)
- typedef \_ios\_Seekdir [seekdir](#)

- typedef \_CharT [char\\_type](#)
- typedef \_Traits::int\_type [int\\_type](#)
- typedef \_Traits::pos\_type [pos\\_type](#)
- typedef \_Traits::off\_type [off\\_type](#)
- typedef \_Traits [traits\\_type](#)

- typedef [ctype](#)< \_CharT > [\\_\\_ctype\\_type](#)
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_get\\_type](#)

## Public Member Functions

- [basic\\_ios](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sb](#))
  - virtual [~basic\\_ios](#) ()
  - const [locale](#) & [\\_M\\_getloc](#) () const
  - void [\\_M\\_setstate](#) ([iostate](#) [\\_\\_state](#))
  - bool [bad](#) () const
  - void [clear](#) ([iostate](#) [\\_\\_state](#)=[goodbit](#))
  - [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &[\\_\\_rhs](#))
  - bool [eof](#) () const
  - [iostate](#) [exceptions](#) () const
  - void [exceptions](#) ([iostate](#) [\\_\\_except](#))
  - bool [fail](#) () const
  - [char\\_type](#) [fill](#) () const
  - [char\\_type](#) [fill](#) ([char\\_type](#) [\\_\\_ch](#))
  - [fmtflags](#) [flags](#) () const
  - [fmtflags](#) [flags](#) ([fmtflags](#) [\\_\\_fmtfl](#))
  - [locale](#) [getloc](#) () const
  - bool [good](#) () const
  - [locale](#) [imbue](#) (const [locale](#) &[\\_\\_loc](#))
  - long & [iword](#) (int [\\_\\_ix](#))
  - [char](#) [narrow](#) ([char\\_type](#) [\\_\\_c](#), [char](#) [\\_\\_dfault](#)) const
  - [streamsize](#) [precision](#) () const
  - [streamsize](#) [precision](#) ([streamsize](#) [\\_\\_prec](#))
  - void \*& [pword](#) (int [\\_\\_ix](#))
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) () const
  - [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* [rdbuf](#) ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_sb](#))
  - [iostate](#) [rdstate](#) () const
  - void [register\\_callback](#) ([event\\_callback](#) [\\_\\_fn](#), int [\\_\\_index](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) [\\_\\_fmtfl](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) [\\_\\_fmtfl](#), [fmtflags](#) [\\_\\_mask](#))
  - void [setstate](#) ([iostate](#) [\\_\\_state](#))
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) () const
  - [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \* [tie](#) ([basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > \*[\\_\\_tiestr](#))
  - void [unsetf](#) ([fmtflags](#) [\\_\\_mask](#))
  - [char\\_type](#) [widen](#) ([char](#) [\\_\\_c](#)) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) [\\_\\_wide](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool [\\_\\_sync](#)=true)
- static int [xalloc](#) () throw ()

## Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

## Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- `basic_ios` ()
- `basic_ios` (const `basic_ios` &)=delete
- void `_M_cache_locale` (const `locale` &\_\_loc)
- void `_M_call_callbacks` (`event` \_\_ev) throw ()
- void `_M_dispose_callbacks` (void) throw ()
- `_Words` & `_M_grow_words` (int \_\_index, bool \_\_iword)
- void `_M_init` () throw ()
- void `_M_move` (`ios_base` &) noexcept
- void `_M_swap` (`ios_base` & \_\_rhs) noexcept
- void `init` (`basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)
- void `move` (`basic_ios` & \_\_rhs)
- void `move` (`basic_ios` && \_\_rhs)
- `basic_ios` & `operator=` (const `basic_ios` &)=delete
- void `set_rdbuf` (`basic_streambuf`< `_CharT`, `_Traits` > \* \_\_sb)
- void `swap` (`basic_ios` & \_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 4.297.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ios< _CharT, _Traits >
```

Template class `basic_ios`, virtual base class for all stream classes.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

Definition at line 77 of file `iosfwd`.

## 4.297.2 Member Typedef Documentation

4.297.2.1 `__ctype_type`

```
template<typename _CharT, typename _Traits>
typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type
```

These are non-standard types.

Definition at line 87 of file `basic_ios.h`.



## 4.297.2.2 \_\_num\_get\_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type
```

These are non-standard types.

Definition at line 91 of file basic\_ios.h.

## 4.297.2.3 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type
```

These are non-standard types.

Definition at line 89 of file basic\_ios.h.

## 4.297.2.4 char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios<_CharT, _Traits>::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.




Definition at line 76 of file basic\_ios.h.

## 4.297.2.5 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                                                                                                      |                                                        |
|------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
|  <code>__e</code> | One of the members of the event enum.                  |
|  <code>__b</code> | Reference to the ios_base object.                      |
|  <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.297.2.6 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.297.2.7 int\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 77 of file basic\_ios.h.

#### 4.297.2.8 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 416 of file ios\_base.h.

#### 4.297.2.9 off\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 79 of file basic\_ios.h.

#### 4.297.2.10 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.297.2.11 pos\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 78 of file `basic_ios.h`.

#### 4.297.2.12 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

#### 4.297.2.13 traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 80 of file basic\_ios.h.

#### 4.297.3 Member Enumeration Documentation

##### 4.297.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

erase\_event is used during ~ios() and copyfmt(). imbue\_event is used during imbue(). copyfmt\_event is used during copyfmt().

Definition at line 512 of file ios\_base.h.

#### 4.297.4 Constructor & Destructor Documentation

##### 4.297.4.1 basic\_ios() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios (
 basic_streambuf< _CharT, _Traits > * __sb) [inline], [explicit]
```

Constructor performs initialization.

The parameter is passed by derived streams.

Definition at line 270 of file basic\_ios.h.

#### 4.297.4.2 ~basic\_ios()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ios< _CharT, _Traits >::~~basic_ios () [inline], [virtual]
```

Empty.

The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

Definition at line 282 of file basic\_ios.h.

#### 4.297.4.3 basic\_ios() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios () [inline], [protected]
```

Empty.

The default constructor does nothing and is not normally accessible to users.

Definition at line 460 of file basic\_ios.h.

### 4.297.5 Member Function Documentation

#### 4.297.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 804 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

## 4.297.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline]
```

Fast error checking.

## Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

## 4.297.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit)
```

[Re]sets the error state.

## Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 4.297.5.4 copyfmt()

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs)
```

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

**4.297.5.5 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

**4.297.5.6 exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

**4.297.5.7 exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline]
```

Throwing exceptions on errors.



## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.297.5.8 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

## 4.297.5.9 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline]
```

Retrieves the *empty* character.

## Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.297.5.10** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and  $Q < P$ . It defaults to a space ( ' ' ) in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.297.5.11** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**4.297.5.12** `flags()` [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file `ios_base.h`.

## 4.297.5.13 getloc()

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, `std::money_put<_CharT, _Outiter>::do_put()`, and `std::ws()`.

## 4.297.5.14 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

## 4.297.5.15 imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc)
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**4.297.5.16 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**4.297.5.17 iword()**

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

**4.297.5.18 narrow()**

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
 char_type __c,
 char __dfault) const [inline]
```

Squeezes characters.

**Parameters**

|                       |                          |
|-----------------------|--------------------------|
| <code>__c</code>      | The character to narrow. |
| <code>__dfault</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, dfault)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file `basic_ios.h`.

**4.297.5.19 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios<_CharT, _Traits>::operator bool () const [inline], [explicit]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

#### 4.297.5.20 `operator!()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

#### 4.297.5.21 `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 4.297.5.22 `precision()` [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

##### Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

##### Returns

The previous value of `precision()`.

Definition at line 728 of file `ios_base.h`.

## 4.297.5.23 pword()

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file ios\_base.h.

## 4.297.5.24 rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf () const [inline]
```

Accessing the underlying buffer.

## Returns

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file basic\_ios.h.

Referenced by std::ws().

## 4.297.5.25 rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits>* __sb)
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.297.5.26 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

**4.297.5.27 `register_callback()`**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.



## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

## 4.297.5.28 setf() [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

## 4.297.5.29 setf() [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

## Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

#### 4.297.5.30 `setstate()`

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline]
```

Sets additional flags in the error state.

##### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

#### 4.297.5.31 `sync_with_stdio()`

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 4.297.5.32 `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.297.5.33 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.297.5.34 unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.297.5.35** `widen()`

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.297.5.36** `width()` [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.297.5.37** `width()` [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of `width()`.

Definition at line 751 of file `ios_base.h`.

## 4.297.5.38 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 4.297.6 Member Data Documentation

## 4.297.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.297.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file ios\_base.h.

#### 4.297.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file ios\_base.h.

#### 4.297.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 4.297.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 399 of file ios\_base.h.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.297.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.297.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.297.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_↵put()`, and `std::noboolalpha()`.

#### 4.297.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.297.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by `std::dec()`.

**4.297.6.11 end**

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

**4.297.6.12 eofbit**

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

**4.297.6.13 failbit**

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry()↵::sentry().

**4.297.6.14 fixed**

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().



## 4.297.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

## 4.297.6.16 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

## 4.297.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<char>::operator<<()`.

## 4.297.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 4.297.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.297.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

#### 4.297.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.297.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 4.297.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

#### 4.297.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

#### 4.297.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put(), std::noshowbase(), and std::showbase().

#### 4.297.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.297.6.27 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.297.6.28 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

**4.297.6.29 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

**4.297.6.30 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

## 4.297.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

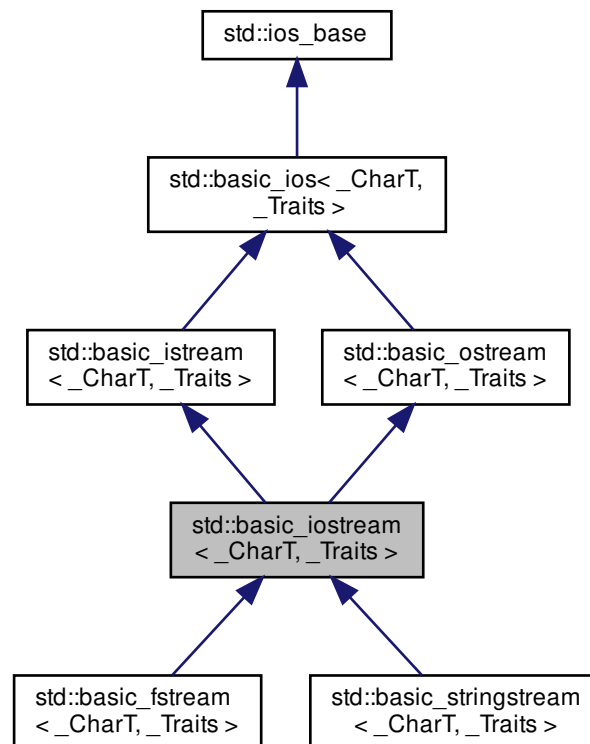
Referenced by std::num\_put< \_CharT, \_Outlter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [basic\\_ios.h](#)
- [basic\\_ios.tcc](#)

## 4.298 std::basic\_iostream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_iostream< \_CharT, \_Traits >:



## Public Types

- typedef `ctype`< `_CharT` > `__ctype_type`
  - typedef `ctype`< `_CharT` > `__ctype_type`
  - typedef `basic_ios`< `_CharT`, `_Traits` > `__ios_type`
  - typedef `basic_ios`< `_CharT`, `_Traits` > `__ios_type`
  - typedef `basic_istream`< `_CharT`, `_Traits` > `__istream_type`
  - typedef `num_get`< `_CharT`, `istreambuf_iterator`< `_CharT`, `_Traits` > > `__num_get_type`
  - typedef `num_put`< `_CharT`, `ostreambuf_iterator`< `_CharT`, `_Traits` > > `__num_put_type`
  - typedef `basic_ostream`< `_CharT`, `_Traits` > `__ostream_type`
  - typedef `basic_streambuf`< `_CharT`, `_Traits` > `__streambuf_type`
  - typedef `basic_streambuf`< `_CharT`, `_Traits` > `__streambuf_type`
  - typedef int `io_state` `_GLIBCXX_DEPRECATED_SUGGEST`("std::iostate")
  - typedef int `open_mode` `_GLIBCXX_DEPRECATED_SUGGEST`("std::openmode")
  - typedef int `seek_dir` `_GLIBCXX_DEPRECATED_SUGGEST`("std::seekdir")
  - typedef `std::streampos` `streampos` `_GLIBCXX_DEPRECATED_SUGGEST`("std::streampos")
  - typedef `std::streamoff` `streamoff` `_GLIBCXX_DEPRECATED_SUGGEST`("std::streamoff")
  - typedef `_CharT` `char_type`
  - enum `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
  - typedef void(\* `event_callback`) (`event` \_\_e, `ios_base` &\_\_b, int \_\_i)
  - typedef `_ios_Fmtflags` `fmtflags`
  - typedef `_Traits::int_type` `int_type`
  - typedef `_ios_iostate` `iostate`
  - typedef `_Traits::off_type` `off_type`
  - typedef `_ios_Openmode` `openmode`
  - typedef `_Traits::pos_type` `pos_type`
  - typedef `_ios_Seekdir` `seekdir`
  - typedef `_Traits` `traits_type`
- 
- typedef `num_put`< `_CharT`, `ostreambuf_iterator`< `_CharT`, `_Traits` > > `__num_put_type`

## Public Member Functions

- `basic_istream` (`basic_streambuf`< `_CharT`, `_Traits` > \*\_\_sb)
- virtual `~basic_istream` ()
- template<typename `_ValueT` >  
`basic_istream`< `_CharT`, `_Traits` > & `_M_extract` (`_ValueT` &\_\_v)
- const `locale` & `_M_getloc` () const
- template<typename `_ValueT` >  
`basic_ostream`< `_CharT`, `_Traits` > & `_M_insert` (`_ValueT` \_\_v)
- void `_M_setstate` (`iostate` \_\_state)
- bool `bad` () const
- void `clear` (`iostate` \_\_state=`goodbit`)
- `basic_ios` & `copyfmt` (const `basic_ios` &\_\_rhs)
- bool `eof` () const
- `iostate` `exceptions` () const
- void `exceptions` (`iostate` \_\_except)

- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- fmtflags [flags](#) () const
- fmtflags [flags](#) (fmtflags \_\_fmtfl)
- \_\_ostream\_type & [flush](#) ()
- streamsize [gcount](#) () const
- template<>  
  [basic\\_istream](#)< char > & [getline](#) (char\_type \* \_\_s, streamsize \_\_n, char\_type \_\_delim)
- template<>  
  [basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \* \_\_s, streamsize \_\_n, char\_type \_\_delim)
- locale [getloc](#) () const
- bool [good](#) () const
- template<>  
  [basic\\_istream](#)< char > & [ignore](#) (streamsize \_\_n)
- template<>  
  [basic\\_istream](#)< char > & [ignore](#) (streamsize \_\_n, int\_type \_\_delim)
- template<>  
  [basic\\_istream](#)< wchar\_t > & [ignore](#) (streamsize \_\_n)
- template<>  
  [basic\\_istream](#)< wchar\_t > & [ignore](#) (streamsize \_\_n, int\_type \_\_delim)
- locale [imbue](#) (const locale & \_\_loc)
- long & [iword](#) (int \_\_ix)
- char [narrow](#) (char\_type \_\_c, char \_\_dfault) const
- \_\_ostream\_type & [operator<<](#) (const void \* \_\_p)
- \_\_ostream\_type & [operator<<](#) (\_\_streambuf\_type \* \_\_sb)
- \_\_istream\_type & [operator>>](#) (void \*& \_\_p)
- \_\_istream\_type & [operator>>](#) (\_\_streambuf\_type \* \_\_sb)
- streamsize [precision](#) () const
- streamsize [precision](#) (streamsize \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) () const
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* [rdbuf](#) ([basic\\_streambuf](#)< \_CharT, \_Traits > \* \_\_sb)
- [iostate](#) [rdstate](#) () const
- void [register\\_callback](#) (event\_callback \_\_fn, int \_\_index)
- \_\_ostream\_type & [seekp](#) (pos\_type)
- \_\_ostream\_type & [seekp](#) (off\_type, ios\_base::seekdir)
- fmtflags [setf](#) (fmtflags \_\_fmtfl)
- fmtflags [setf](#) (fmtflags \_\_fmtfl, fmtflags \_\_mask)
- void [setstate](#) (iostate \_\_state)
- pos\_type [tellp](#) ()
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
- [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
- void [unsetf](#) (fmtflags \_\_mask)
- char\_type [widen](#) (char \_\_c) const
- streamsize [width](#) () const
- streamsize [width](#) (streamsize \_\_wide)

- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
- `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
- `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `false`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`
  - `__istream_type & operator>> (long double &__f)`

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to `true`. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type get ()`
- `__istream_type & get (char_type &__c)`
- `__istream_type & get (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & get (char_type *__s, streamsize __n)`
- `__istream_type & get (__streambuf_type &__sb, char_type __delim)`
- `__istream_type & get (__streambuf_type &__sb)`
- `__istream_type & getline (char_type *__s, streamsize __n, char_type __delim)`
- `__istream_type & getline (char_type *__s, streamsize __n)`
- `__istream_type & ignore (streamsize __n, int_type __delim)`
- `__istream_type & ignore (streamsize __n)`
- `__istream_type & ignore ()`
- `int_type peek ()`



- [\\_\\_istream\\_type](#) & [read](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - int [sync](#) ()
  - pos\_type [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - bool [operator!](#) () const
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ( [\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ( [\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the *operator<<* functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

## Protected Member Functions

- **basic\_istream** (const [basic\\_istream](#) &)=delete
- **basic\_istream** ([basic\\_istream](#) && \_\_rhs)
- void **\_M\_cache\_locale** (const [locale](#) & \_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_istream\\_type](#) & **\_M\_extract** (\_ValueT & \_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  [\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) & \_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void **move** ([basic\\_ios](#) & \_\_rhs)
- void **move** ([basic\\_ios](#) && \_\_rhs)
- [basic\\_istream](#) & **operator=** (const [basic\\_istream](#) &)=delete
- [basic\\_istream](#) & **operator=** ([basic\\_istream](#) && \_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)<\_CharT, \_Traits> \* \_\_sb)
- void **swap** ([basic\\_ostream](#) & \_\_rhs)
- void **swap** ([basic\\_ios](#) & \_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) & \_\_rhs)
- void **swap** ([basic\\_istream](#) & \_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)<\_CharT, \_Traits> \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)<\_CharT, \_Traits> \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.298.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream<_CharT, _Traits>
```

Template class basic\_istream.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class multiply inherits from the input and output stream classes simply to provide a single interface.

Definition at line 89 of file `iosfwd`.

### 4.298.2 Member Typedef Documentation

#### 4.298.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

#### 4.298.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.298.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.298.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 416 of file `ios_base.h`.

#### 4.298.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.298.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.298.3 Member Enumeration Documentation

#### 4.298.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

## 4.298.4 Constructor &amp; Destructor Documentation

## 4.298.4.1 basic\_iostream()

```
template<typename _CharT, typename _Traits>
std::basic_iostream< _CharT, _Traits >::basic_iostream (
 basic_streambuf< _CharT, _Traits > * __sb) [inline], [explicit]
```

Constructor does nothing.

Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

Definition at line 849 of file istream.

## 4.298.4.2 ~basic\_iostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_iostream< _CharT, _Traits >::~~basic_iostream () [inline], [virtual]
```

Destructor does nothing.

Definition at line 856 of file istream.

## 4.298.5 Member Function Documentation

## 4.298.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

## Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 804 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

## 4.298.5.2 \_M\_write()

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>_↵<br/>_s</code> | The array to insert.                    |
| <code>_↵<br/>_n</code> | Maximum number of characters to insert. |

Definition at line 317 of file ostream.

**4.298.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.298.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >↵  
::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.



## 4.298.5.5 copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios<_CharT, _Traits> & std::basic_ios<_CharT, _Traits>::copyfmt (
 const basic_ios<_CharT, _Traits> & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

## Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

## 4.298.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::eof () const [inline], [inherited]
```

Fast error checking.

## Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

## 4.298.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios<_CharT, _Traits>::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

## Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::copyfmt().

#### 4.298.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

##### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

#### 4.298.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

## 4.298.5.10 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file basic\_ios.h.

Referenced by std::basic\_ios<char, \_Traits>::copyfmt(), std::basic\_ios<char, \_Traits>::fill(), and std::operator<<().

## 4.298.5.11 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

## 4.298.5.12 flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**4.298.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

**4.298.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

## 4.298.5.15 gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

## Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

## 4.298.5.16 get() [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

## 4.298.5.17 get() [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

**4.298.5.18 `get()`** [3/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file `istream.tcc`.

## 4.298.5.19 get() [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

## Returns

\*this

Returns `get(__s, __n, widen("\n"))`.

Definition at line 354 of file istream.

## 4.298.5.20 get() [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

## Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

## Returns

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF

- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file `istream.tcc`.

#### 4.298.5.21 `get()` [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

##### Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

##### Returns

`*this`

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file `istream`.

#### 4.298.5.22 `getline()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

##### Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |



**Returns**

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file istream.tcc.

**4.298.5.23 getline()** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

\*this

Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file istream.

**4.298.5.24** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**4.298.5.25** `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIt >::do_put()`, and `std::ws()`.

**4.298.5.26** `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

**4.298.5.27** `ignore()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

## Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**4.298.5.28** `ignore()` [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

## Returns

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

**4.298.5.29 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**4.298.5.30 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

**4.298.5.31 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.


Referenced by std::basic\_ios< char, \_Traits >::basic\_ios().

#### 4.298.5.32 iword()

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

##### Parameters

|                                                                                                  |                       |
|--------------------------------------------------------------------------------------------------|-----------------------|
| <br><i>__ix</i> | Index into the array. |
|--------------------------------------------------------------------------------------------------|-----------------------|

##### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file ios\_base.h.

#### 4.298.5.33 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                  |                          |
|------------------|--------------------------|
| <i>__c</i>       | The character to narrow. |
| <i>__default</i> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

**4.298.5.34 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**4.298.5.35 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**4.298.5.36 operator<<() [1/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

## 4.298.5.37 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 __ios_type &(*)(__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

## 4.298.5.38 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

## 4.298.5.39 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file `ostream`.

**4.298.5.40** `operator<<()` [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.298.5.41** `operator<<()` [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file `ostream`.



## 4.298.5.42 operator&lt;&lt;() [7/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

**4.298.5.43 operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 181 of file `ostream`.

**4.298.5.44 operator<<() [9/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                                                                                                    |                                      |
|----------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|----------------------------------------------------------------------------------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file `ostream.tcc`.

## 4.298.5.45 operator&lt;&lt;() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|------------------------------------------------------------------------------------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file `ostream`.

## 4.298.5.46 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

**4.298.5.47 operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**4.298.5.48 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file ostream.

## 4.298.5.49 operator&lt;&lt;() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 4.298.5.50 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**4.298.5.51 operator<<() [16/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| $\_ \leftarrow$ | A variable of pointer type. |
| $\_p$           |                             |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file ostream.

**4.298.5.52 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

4.298.5.53 `operator>>()` [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

4.298.5.54 `operator>>()` [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**4.298.5.55** `operator>>()` [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**4.298.5.56** `operator>>()` [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

**4.298.5.57** `operator>>()` [5/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|



**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**4.298.5.58 operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

**4.298.5.59 operator>>() [7/17]**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**4.298.5.60 operator>>() [8/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                     |                                      |
|---------------------|--------------------------------------|
| $\leftarrow$<br>__n | A variable of builtin integral type. |
|---------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.298.5.61 operator>>() [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                     |                                      |
|---------------------|--------------------------------------|
| $\leftarrow$<br>__n | A variable of builtin integral type. |
|---------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.298.5.62 operator>>() [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**4.298.5.63 operator>>() [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**4.298.5.64 operator>>() [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**4.298.5.65 operator>>() [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                                           |                                            |
|---------------------------------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_ \leftarrow$<br>$\leftarrow$<br>$\_ \leftarrow$<br>$f$ | A variable of builtin floating point type. |
|---------------------------------------------------------------------------|--------------------------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**4.298.5.66 operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**4.298.5.67 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**4.298.5.68 operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | A variable of pointer type. |
|------------------|-----------------------------|

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.298.5.69 operator>>()** [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, `failbit` is set.

Definition at line 212 of file `istream.tcc`.

#### 4.298.5.70 peek()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

##### Returns

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

#### 4.298.5.71 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 4.298.5.72 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

Definition at line 728 of file `ios_base.h`.

**4.298.5.73 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

`*this`

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file `ostream.tcc`.

**4.298.5.74 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.



**Parameters**

|                 |                                                   |
|-----------------|---------------------------------------------------|
| <code>_C</code> | The character to push back into the input stream. |
|-----------------|---------------------------------------------------|

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**4.298.5.75 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file `ios_base.h`.

**4.298.5.76** `rdbuf()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::rdbuf () const [inline],
[inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file `basic_ios.h`.

Referenced by `std::ws()`.

**4.298.5.77** `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (
 basic_streambuf<_CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

## 4.298.5.78 rdstate()

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See std::ios\_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, ↵  
\_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream<  
char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 4.298.5.79 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

## Parameters

|          |                                        |
|----------|----------------------------------------|
| ↵<br>__s | A character array.                     |
| ↵<br>__n | Maximum number of characters to store. |

## Returns

\*this

If the stream state is good ( ) , extracts characters and stores them into \_\_s until one of the following happens:

- \_\_n characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to failbit|eofbit.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file istream.tcc.

4.298.5.80 `readsome()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits>::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rddbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file `istream.tcc`.

4.298.5.81 `register_callback()`

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 4.298.5.82 seekg() [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets `failbit`.

##### Note

This function first clears `eofbit`. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

#### 4.298.5.83 seekg() [2/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

##### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

##### Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

#### Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

#### 4.298.5.84 `seekp()` [1/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

#### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

#### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

#### 4.298.5.85 `seekp()` [2/2]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

#### Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

**4.298.5.86 setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**4.298.5.87 setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

**4.298.5.88 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**4.298.5.89 sync()**

```
template<typename _CharT , typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.



## 4.298.5.90 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

## 4.298.5.91 tellg()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

## Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

## 4.298.5.92 tellp()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
[inherited]
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**4.298.5.93** `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`.

**4.298.5.94** `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits>::tie (
 basic_ostream<_CharT, _Traits> * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.298.5.95** `unget()`

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**4.298.5.96 unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.298.5.97 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

**Returns the result of**

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.298.5.98 width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::num_put<_CharT, _OutIter>::do_put()`.

**4.298.5.99 width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.



Definition at line 751 of file ios\_base.h.

#### 4.298.5.100 write()

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

##### Parameters

|                                                                                                    |                                         |
|----------------------------------------------------------------------------------------------------|-----------------------------------------|
|  <code>__s</code> | The array to insert.                    |
|  <code>__n</code> | Maximum number of characters to insert. |

##### Returns

`*this`

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 4.298.5.101 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

## 4.298.6 Member Data Documentation

### 4.298.6.1 `_M_gcount`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, and `std::basic_istream< char >::unget()`.

### 4.298.6.2 `adjustfield`

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

### 4.298.6.3 `app`

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

### 4.298.6.4 `ate`

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.298.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 4.298.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 399 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), std::oct(), and std::basic\_ostream< char >::operator<<().

#### 4.298.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.298.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.298.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_←put()`, and `std::noboolalpha()`.

#### 4.298.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.298.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.298.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.



#### 4.298.6.13 eofbit

```
const ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_date()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_time()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::eof()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_istream< char >::unget()`, and `std::ws()`.

#### 4.298.6.14 failbit

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry()`.

#### 4.298.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::hexfloat()`.

#### 4.298.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.298.6.17 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsome()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

#### 4.298.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.298.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 4.298.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.298.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 4.298.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.298.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

**4.298.6.24 right**

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file `ios_base.h`.

Referenced by `std::right()`.

**4.298.6.25 scientific**

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::hexfloat()`, and `std::scientific()`.

**4.298.6.26 showbase**

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.

**4.298.6.27 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file `ios_base.h`.

Referenced by `std::noshowpoint()`, and `std::showpoint()`.

#### 4.298.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.298.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.298.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.298.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().



- typedef \_ios\_Fmtflags [fmtflags](#)
  - typedef \_Traits::int\_type **int\_type**
  - typedef \_ios\_istate [iostate](#)
  - typedef \_Traits::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef \_Traits::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > [\\_\\_num\\_put\\_type](#)

#### Public Member Functions

- [basic\\_istream](#) ([\\_\\_streambuf\\_type](#) \*\_\_sb)
- virtual [~basic\\_istream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **M\_extract** (\_ValueT &\_\_v)
- const [locale](#) & [M\\_getloc](#) () const
- void **M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) &\_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- char\_type [fill](#) () const
- char\_type [fill](#) (char\_type \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [streamsize](#) [gcount](#) () const
- template<>  
[basic\\_istream](#)< char > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- template<>  
[basic\\_istream](#)< wchar\_t > & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
- [locale](#) [getloc](#) () const
- bool [good](#) () const
- template<>  
[basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< char > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- template<>  
[basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n)
- template<>  
[basic\\_istream](#)< wchar\_t > & **ignore** ([streamsize](#) \_\_n, int\_type \_\_delim)
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc)
- long & [iword](#) (int \_\_ix)

- `char narrow (char_type __c, char __default) const`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf<_CharT, _Traits> * rdbuf () const`
  - `basic_streambuf<_CharT, _Traits> * rdbuf (basic_streambuf<_CharT, _Traits> *__sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `basic_ostream<_CharT, _Traits> * tie () const`
  - `basic_ostream<_CharT, _Traits> * tie (basic_ostream<_CharT, _Traits> *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(__pf)(__ios_type &))`
  - `__istream_type & operator>> (ios_base &(__pf)(ios_base &))`

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `__istream_type & operator>> (bool &__n)`
  - `__istream_type & operator>> (short &__n)`
  - `__istream_type & operator>> (unsigned short &__n)`
  - `__istream_type & operator>> (int &__n)`
  - `__istream_type & operator>> (unsigned int &__n)`
  - `__istream_type & operator>> (long &__n)`
  - `__istream_type & operator>> (unsigned long &__n)`
  - `__istream_type & operator>> (long long &__n)`
  - `__istream_type & operator>> (unsigned long long &__n)`
- 
- `__istream_type & operator>> (float &__f)`
  - `__istream_type & operator>> (double &__f)`



- [\\_\\_istream\\_type](#) & [operator>>](#) (long double &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the `sentry` documentation for more.

If the `sentry` status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type &\_\_c)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) &\_\_sb)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n, char\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [getline](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, int\_type \_\_delim)
  - [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [ignore](#) ()
  - `int_type` [peek](#) ()
  - [\\_\\_istream\\_type](#) & [read](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [streamsize](#) [readsome](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
  - [\\_\\_istream\\_type](#) & [unget](#) ()
  - `int` [sync](#) ()
  - `pos_type` [tellg](#) ()
  - [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
  - [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, [ios\\_base::seekdir](#))
- 
- [operator bool](#) () const
  - `bool` [operator!](#) () const

### Static Public Member Functions

- static `bool` [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static `int` [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iostate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iostate](#) [eofbit](#)
- static const [iostate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iostate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- **[basic\\_istream](#)** (const [basic\\_istream](#) &)=delete
- **[basic\\_istream](#)** ([basic\\_istream](#) &&\_\_rhs)
- void **[\\_M\\_cache\\_locale](#)** (const [locale](#) &\_\_loc)
- void **[\\_M\\_call\\_callbacks](#)** ([event](#) \_\_ev) throw ()
- void **[\\_M\\_dispose\\_callbacks](#)** (void) throw ()
- template<typename [\\_ValueT](#) >  
  [\\_istream\\_type](#) & **[\\_M\\_extract](#)** ([\\_ValueT](#) &\_\_v)
- [\\_Words](#) & **[\\_M\\_grow\\_words](#)** (int \_\_index, bool \_\_iword)
- void **[\\_M\\_init](#)** () throw ()

- void **\_M\_move** (ios\_base &) noexcept
- void **\_M\_swap** (ios\_base & \_\_rhs) noexcept
- void **init** (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- void **move** (basic\_ios & \_\_rhs)
- void **move** (basic\_ios && \_\_rhs)
- **basic\_istream** & **operator=** (const **basic\_istream** &)=delete
- **basic\_istream** & **operator=** (**basic\_istream** && \_\_rhs)
- void **set\_rdbuf** (basic\_streambuf< \_CharT, \_Traits > \* \_\_sb)
- void **swap** (basic\_ios & \_\_rhs) noexcept
- void **swap** (basic\_istream & \_\_rhs)

#### Protected Attributes

- \_Callback\_list \* **\_M\_callbacks**
- const \_\_ctype\_type \* **\_M\_ctype**
- iostate **\_M\_exception**
- char\_type **\_M\_fill**
- bool **\_M\_fill\_init**
- fmtflags **\_M\_flags**
- streamsize **\_M\_gcount**
- locale **\_M\_ios\_locale**
- \_Words **\_M\_local\_word** [\_S\_local\_word\_size]
- const \_\_num\_get\_type \* **\_M\_num\_get**
- const \_\_num\_put\_type \* **\_M\_num\_put**
- streamsize **\_M\_precision**
- basic\_streambuf< \_CharT, \_Traits > \* **\_M\_streambuf**
- iostate **\_M\_streambuf\_state**
- basic\_ostream< \_CharT, \_Traits > \* **\_M\_tie**
- streamsize **\_M\_width**
- \_Words \* **\_M\_word**
- int **\_M\_word\_size**
- \_Words **\_M\_word\_zero**

#### Friends

- class **sentry**

#### 4.299.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >
```

Template class basic\_istream.

#### Template Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                   |
| <code>_Traits</code> | Traits for character type, defaults to char_traits<_CharT>. |

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual input.

Definition at line 83 of file `iosfwd`.

## 4.299.2 Member Typedef Documentation

### 4.299.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 4.299.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

#### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

### 4.299.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.299.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 416 of file `ios_base.h`.

#### 4.299.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.299.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.299.3 Member Enumeration Documentation

#### 4.299.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

## 4.299.4 Constructor &amp; Destructor Documentation

## 4.299.4.1 basic\_istream()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
 __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 93 of file istream.

## 4.299.4.2 ~basic\_istream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~~basic_istream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 103 of file istream.

## 4.299.5 Member Function Documentation

## 4.299.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

Definition at line 804 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_Inlter >::do\_get(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::time\_get< \_CharT, \_Inlter >::do\_get\_date(), std::time\_get< \_CharT, \_Inlter >::do\_get\_monthname(), std::time\_get< \_CharT, \_Inlter >::do\_get\_time(), std::time\_get< \_CharT, \_Inlter >::do\_get\_weekday(), std::time\_put< \_CharT, \_Outlter >::do\_put(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::time\_get< \_CharT, \_Inlter >::get(), and std::time\_put< \_CharT, \_Outlter >::put().

#### 4.299.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

#### 4.299.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

##### Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See `std::ios_base::iostate` for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by `std::basic_ios< char, _Traits >::exceptions()`, `std::__detail::operator>>()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

#### 4.299.5.4 copyfmt()

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|



**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

**4.299.5.5 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file `basic_ios.h`.

**4.299.5.6 exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

**4.299.5.7 exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

### Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

#### 4.299.5.8 `fail()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

### Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

#### 4.299.5.9 `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

### Returns

The current fill character.

It defaults to a space (' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

## 4.299.5.10 fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios<_CharT, _Traits>::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

## Parameters

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

## Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file basic\_ios.h.

## 4.299.5.11 flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

## Returns

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits>::copyfmt(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_OutIter>::do\_put(), std::basic\_ostream< char>::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

## 4.299.5.12 flags() [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file `ios_base.h`.

**4.299.5.13 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

**4.299.5.14 get()** [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void)
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 244 of file `istream.tcc`.

**4.299.5.15 get()** [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c)
```

Simple extraction.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

## Returns

\*this

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

4.299.5.16 `get()` [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

\*this

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

**4.299.5.17 get()** [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**4.299.5.18 get()** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim)
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

**4.299.5.19 get()** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**4.299.5.20 getline()** [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

String extraction.

**Parameters**

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

**Returns**

`*this`

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file `istream.tcc`.

**4.299.5.21 `getline()`** [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`



Returns `getline(__s, __n, widen("\n"))`.

Definition at line 427 of file `istream`.

#### 4.299.5.22 `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim)
```

Explicit specialization declarations, defined in `src/istream.cc`.

#### 4.299.5.23 `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

#### 4.299.5.24 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.299.5.25 `ignore()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim)
```

Discarding characters.

**Parameters**

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

**Returns**

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file `istream.tcc`.

**4.299.5.26 ignore()** [2/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n)
```

Simple extraction.

**Returns**

A character, or `eof()`.

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file `istream.tcc`.

## 4.299.5.27 ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::ignore (
 void)
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

## 4.299.5.28 imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

## 4.299.5.29 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
 basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

#### 4.299.5.30 `iword()`

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

##### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

#### 4.299.5.31 `narrow()`

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 430 of file `basic_ios.h`.

**4.299.5.32 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

**4.299.5.33 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 125 of file `basic_ios.h`.

**4.299.5.34 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

**4.299.5.35 operator>>()** [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

**4.299.5.36 operator>>()** [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

**4.299.5.37 operator>>()** [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

## 4.299.5.38 operator&gt;&gt;() [5/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n)
```

Integer arithmetic extractors.

## Parameters

|                     |                                      |
|---------------------|--------------------------------------|
| $\leftarrow$<br>__n | A variable of builtin integral type. |
|---------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 122 of file istream.tcc.

## 4.299.5.39 operator&gt;&gt;() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline]
```

Integer arithmetic extractors.

## Parameters

|                     |                                      |
|---------------------|--------------------------------------|
| $\leftarrow$<br>__n | A variable of builtin integral type. |
|---------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the num\_get facet) to parse the input data.

Definition at line 175 of file istream.

**4.299.5.40** `operator>>()` [7/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n)
```

Integer arithmetic extractors.



## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

## 4.299.5.41 operator&gt;&gt;() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

## 4.299.5.42 operator&gt;&gt;() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                  |                                      |
|----------------------------------|--------------------------------------|
| $\leftarrow$<br><code>__n</code> | A variable of builtin integral type. |
|----------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.299.5.43 `operator>>()` [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline]
```

Integer arithmetic extractors.

**Parameters**

|                                  |                                      |
|----------------------------------|--------------------------------------|
| $\leftarrow$<br><code>__n</code> | A variable of builtin integral type. |
|----------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

**4.299.5.44 `operator>>()` [11/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

## 4.299.5.45 operator&gt;&gt;() [12/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

## 4.299.5.46 operator&gt;&gt;() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**4.299.5.47 operator>>()** [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline]
```

Floating point arithmetic extractors.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

**4.299.5.48 operator>>()** [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline]
```

Floating point arithmetic extractors.

## Parameters

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

## 4.299.5.49 operator&gt;&gt;() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline]
```

Basic arithmetic extractors.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| $\_ \leftarrow$ | A variable of pointer type. |
| $\_p$           |                             |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

## 4.299.5.50 operator&gt;&gt;() [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb)
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

**4.299.5.51 peek()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void)
```

Looking ahead in the stream.

**Returns**

The next character, or `eof()`.

If, after constructing the sentry object, `good()` is `false`, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

**4.299.5.52 precision()** [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**4.299.5.53 precision()** [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

## Returns

The previous value of `precision()`.

Definition at line 728 of file `ios_base.h`.

## 4.299.5.54 putback()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c)
```

Unextracting a single character.

## Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

## Returns

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

## Note

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

## 4.299.5.55 pword()

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>_↵<br/>_ix</code> | Index into the array. |
|-------------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pwd` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file `ios_base.h`.

**4.299.5.56** `rdbuf()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::rdbuf () const [inline],
[inherited]
```

Accessing the underlying buffer.

**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file `basic_ios.h`.

Referenced by `std::ws()`.

**4.299.5.57** `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits > * std::basic_ios<_CharT, _Traits >::rdbuf (
 basic_streambuf<_CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.



## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

## 4.299.5.58 rdbuf()

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdbuf () const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

## 4.299.5.59 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n)
```

Extraction without delimiters.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

`*this`

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

**4.299.5.60 readsome()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the `streambuf`'s buffer, `rdbuf()->in_avail()`, called A here:

- if  $A == -1$ , sets eofbit and extracts no characters
- if  $A == 0$ , extracts no characters
- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

#### 4.299.5.61 register\_callback()

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

##### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 4.299.5.62 seekg() [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos)
```

Changing the current read position.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**4.299.5.63 seekg()** [2/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir)
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**4.299.5.64 setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

**4.299.5.65 setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file ios\_base.h.

**4.299.5.66 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 4.299.5.67 `sync()`

```
template<typename _CharT, typename _Traits>
int std::basic_istream<_CharT, _Traits>::sync (
 void)
```

Synchronizing the stream buffer.

##### Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

##### Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

#### 4.299.5.68 `sync_with_stdio()`

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**4.299.5.69 tellg()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void)
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**4.299.5.70 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.299.5.71 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.299.5.72 `unget()`**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void)
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**4.299.5.73 `unsetf()`**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.



## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

4.299.5.74 `widen()`

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

4.299.5.75 `width()` [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.299.5.76** `width()` [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file `ios_base.h`.

**4.299.5.77** `xalloc()`

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.299.6** Member Data Documentation**4.299.6.1** `_M_gcount`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, and `std::basic_istream< char >::unget()`.

#### 4.299.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.299.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

#### 4.299.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.299.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 4.299.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.299.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.299.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.299.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

#### 4.299.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 4.299.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::dec().

#### 4.299.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.299.6.13 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 4.299.6.14 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry() and std::basic\_ostream< \_CharT, \_Traits >::flush().

#### 4.299.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().

#### 4.299.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 402 of file ios\_base.h.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

#### 4.299.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

#### 4.299.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

#### 4.299.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 4.299.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.299.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

#### 4.299.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.299.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 4.299.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file `ios_base.h`.

Referenced by `std::right()`.

#### 4.299.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::hexfloat()`, and `std::scientific()`.

#### 4.299.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.



#### 4.299.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.299.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.299.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.299.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.299.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

#### 4.299.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

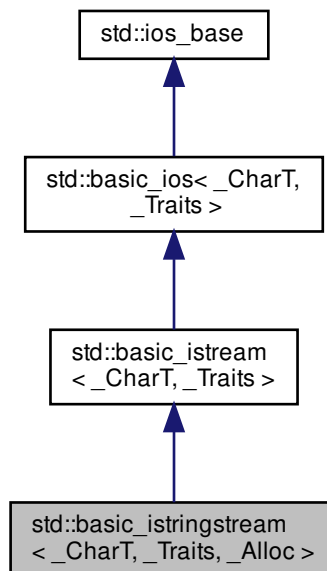
Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [istream](#)
- [istream.tcc](#)

### 4.300 std::basic\_istream< \_CharT, \_Traits, \_Alloc > Class Template Reference

Inheritance diagram for std::basic\_istream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [basic\\_istream](#)< char\_type, traits\_type > **\_\_istream\_type**
  - typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **\_\_string\_type**
  - typedef [basic\\_stringbuf](#)< \_CharT, \_Traits, \_Alloc > **\_\_stringbuf\_type**
  - typedef int io\_state **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::iostate")
  - typedef int open\_mode **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::openmode")
  - typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::seekdir")
  - typedef [std::streampos](#) [streampos](#) **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streampos")
  - typedef [std::streamoff](#) [streamoff](#) **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streamoff")
  - typedef \_Alloc **allocator\_type**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags [fmtflags](#)
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_iostate [iostate](#)
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode [openmode](#)
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir [seekdir](#)
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**

## Public Member Functions

- [basic\\_istream](#) ()
- [basic\\_istream](#) ([ios\\_base::openmode](#) \_\_mode)
- [basic\\_istream](#) (const [\\_\\_string\\_type](#) & \_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#))
- **basic\_istream** (const [basic\\_istream](#) &)=delete
- **basic\_istream** ([basic\\_istream](#) && \_\_rhs)
- [~basic\\_istream](#) ()
- template<typename \_ValueT >  
[basic\\_istream](#)< \_CharT, \_Traits > & **\_M\_extract** (\_ValueT & \_\_v)
- const [locale](#) & **\_M\_getloc** () const
- void **\_M\_setstate** ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)

- `bool fail () const`
  - `char_type fill () const`
  - `char_type fill (char_type __ch)`
  - `fmtflags flags () const`
  - `fmtflags flags (fmtflags __fmtfl)`
  - `streamsize gcount () const`
  - `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `basic_istream & operator= (const basic_istream &)=delete`
  - `basic_istream & operator= (basic_istream &&__rhs)`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `__stringbuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `__string_type str () const`
  - `void str (const __string_type &__s)`
  - `void swap (basic_istream &__rhs)`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`

- [\\_\\_istream\\_type & operator>>](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
- [\\_\\_istream\\_type & operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type & operator>>](#) ([bool](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([short](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([unsigned short](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([int](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([unsigned int](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([long](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([unsigned long](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([long long](#) &\_\_n)
  - [\\_\\_istream\\_type & operator>>](#) ([unsigned long long](#) &\_\_n)
- 
- [\\_\\_istream\\_type & operator>>](#) ([float](#) &\_\_f)
  - [\\_\\_istream\\_type & operator>>](#) ([double](#) &\_\_f)
  - [\\_\\_istream\\_type & operator>>](#) ([long double](#) &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [int\\_type get](#) ()
- [\\_\\_istream\\_type & get](#) ([char\\_type](#) &\_\_c)
- [\\_\\_istream\\_type & get](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type & get](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type & get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type & get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb)
- [\\_\\_istream\\_type & getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n, [char\\_type](#) \_\_delim)
- [\\_\\_istream\\_type & getline](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type & ignore](#) ([streamsize](#) \_\_n, [int\\_type](#) \_\_delim)
- [\\_\\_istream\\_type & ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type & ignore](#) ()
- [int\\_type peek](#) ()
- [\\_\\_istream\\_type & read](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [streamsize readsome](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)

- [\\_\\_istream\\_type](#) & [putback](#) (char\_type \_\_c)
- [\\_\\_istream\\_type](#) & [unget](#) ()
- int [sync](#) ()
- pos\_type [tellg](#) ()
- [\\_\\_istream\\_type](#) & [seekg](#) (pos\_type)
- [\\_\\_istream\\_type](#) & [seekg](#) (off\_type, ios\_base::seekdir)

- [operator bool](#) () const
- bool [operator!](#) () const

#### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

#### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.300.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istream< _CharT, _Traits, _Alloc >
```

Controlling input for std::string.

### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports reading from objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 100 of file `iosfwd`.

## 4.300.2 Member Typedef Documentation

### 4.300.2.1 `__num_put_type`

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

### 4.300.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

|                       |                                                        |
|-----------------------|--------------------------------------------------------|
| <code>↔<br/>_e</code> | One of the members of the event enum.                  |
| <code>↔<br/>_b</code> | Reference to the <code>ios_base</code> object.         |
| <code>↔<br/>_i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.



#### 4.300.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- *boolalpha*
- *dec*
- *fixed*
- *hex*
- *internal*
- *left*
- *oct*
- *right*
- *scientific*
- *showbase*
- *showpoint*
- *showpos*
- *skipws*
- *unitbuf*
- *uppercase*
- *adjustfield*
- *basefield*
- *floatfield*

Definition at line 341 of file *ios\_base.h*.

#### 4.300.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- *badbit*
- *eofbit*
- *failbit*
- *goodbit*

Definition at line 416 of file *ios\_base.h*.

#### 4.300.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.300.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.300.3 Member Enumeration Documentation

#### 4.300.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

## 4.300.4 Constructor &amp; Destructor Documentation

## 4.300.4.1 basic\_istream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using `in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 425 of file `sstream`.

## 4.300.4.2 basic\_istream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::in` is automatically included in `__mode`.

Initializes `sb` using `__mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 442 of file `sstream`.

## 4.300.4.3 basic\_istream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::basic_istream (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

#### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::in` is automatically included in `mode`.

Initializes `sb` using `str` and `mode|in`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 460 of file `sstream`.

#### 4.300.4.4 `~basic_istream()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits, _Alloc >::~~basic_istream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 471 of file `sstream`.

### 4.300.5 Member Function Documentation

#### 4.300.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

#### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

## 4.300.5.2 bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

## Returns

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

## 4.300.5.3 clear()

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

## Parameters

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

## 4.300.5.4 copyfmt()

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

## Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

**4.300.5.5 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

**4.300.5.6 exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

**4.300.5.7 exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.300.5.8 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

## 4.300.5.9 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

## Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.300.5.10** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and  $Q < P$ . It defaults to a space (' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.300.5.11** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**4.300.5.12** `flags()` [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.



## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

## 4.300.5.13 gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const [inline], [inherited]
```

Character counting.

## Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file istream.

## 4.300.5.14 get() [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

## 4.300.5.15 get() [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

`*this`

Tries to extract a character and store it in `__c`. If none are available, sets failbit and returns `traits::eof()`.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file `istream.tcc`.

**4.300.5.16 `get()`** [3/6]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

**Returns**

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

**4.300.5.17 get()** [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

**Parameters**

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file istream.

**4.300.5.18 get()** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

Definition at line 364 of file istream.tcc.

**4.300.5.19** `get()` [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.

**Parameters**

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

**Returns**

\*this

Returns `get(__sb,widen("\n"))`.

Definition at line 387 of file istream.

**4.300.5.20** `getline()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

## Returns

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file istream.tcc.

4.300.5.21 `getline()` [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

## Parameters

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

## Returns

\*this

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

#### 4.300.5.22 `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

#### 4.300.5.23 `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outlter >::do_put()`, and `std::ws()`.

#### 4.300.5.24 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.300.5.25 `ignore()` [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

## Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file istream.tcc.

## 4.300.5.26 ignore() [2/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

## Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns `traits::eof()`.

Definition at line 501 of file istream.tcc.

**4.300.5.27 ignore()** [3/3]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**4.300.5.28 imbue()**

```
template<typename _CharT, typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file basic\_ios.tcc.

**4.300.5.29 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```



All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.


Referenced by std::basic\_ios< char, \_Traits >::basic\_ios().

#### 4.300.5.30 iword()

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

##### Parameters

|                                                                                           |                       |
|-------------------------------------------------------------------------------------------|-----------------------|
| <br>__ix | Index into the array. |
|-------------------------------------------------------------------------------------------|-----------------------|

##### Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file ios\_base.h.

#### 4.300.5.31 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|           |                          |
|-----------|--------------------------|
| __c       | The character to narrow. |
| __default | The character to narrow. |

**Returns**

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c, default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

**4.300.5.32 operator bool()**

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 117 of file `basic_ios.h`.

**4.300.5.33 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**4.300.5.34 operator>>() [1/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

## 4.300.5.35 operator&gt;&gt;() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

## 4.300.5.36 operator&gt;&gt;() [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 131 of file `istream`.

## 4.300.5.37 operator&gt;&gt;() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 168 of file `istream`.

**4.300.5.38** `operator>>()` [5/17]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**4.300.5.39** `operator>>()` [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file `istream`.

## 4.300.5.40 operator&gt;&gt;() [7/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file `istream.tcc`.

**4.300.5.41 `operator>>()`** [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.300.5.42 `operator>>()`** [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↵</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

## 4.300.5.43 operator&gt;&gt;() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↵</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

## 4.300.5.44 operator&gt;&gt;() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

**4.300.5.45 `operator>>()` [12/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|       |                                      |
|-------|--------------------------------------|
| $\_↵$ | A variable of builtin integral type. |
| $\_n$ |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**4.300.5.46 `operator>>()` [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.



## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

## 4.300.5.47 operator&gt;&gt;() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

## 4.300.5.48 operator&gt;&gt;() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                 |                                            |
|-----------------|--------------------------------------------|
| $\leftarrow$    | A variable of builtin floating point type. |
| $\_ \leftarrow$ |                                            |
| $\leftarrow$    |                                            |
| $\_ \leftarrow$ |                                            |
| <i>f</i>        |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

**4.300.5.49 operator>>()** [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| $\_ \leftarrow$ | A variable of pointer type. |
| $\_p$           |                             |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.300.5.50 operator>>()** [17/17]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is NULL, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file istream.tcc.

## 4.300.5.51 peek()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
 void) [inherited]
```

Looking ahead in the stream.

## Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file istream.tcc.

## 4.300.5.52 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

## 4.300.5.53 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

Definition at line 728 of file `ios_base.h`.

**4.300.5.54 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | The character to push back into the input stream. |
|------------------|---------------------------------------------------|

**Returns**

`*this`

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**4.300.5.55 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

## Parameters

|                  |                       |
|------------------|-----------------------|
| <code>_ix</code> | Index into the array. |
|------------------|-----------------------|

## Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file ios\_base.h.

## 4.300.5.56 rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.300.5.57 rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_istream< _CharT, _Traits, _Alloc >::rdbuf () const [inline]
```

Accessing the underlying buffer.

**Returns**

The current basic\_stringbuf buffer.

This hides both signatures of std::basic\_ios::rdbuf().

Definition at line 511 of file sstream.

**4.300.5.58 rdstate()**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See std::ios\_base::iosstate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

Definition at line 137 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::eof(), std::basic\_ios< char, \_Traits >::fail(), std::basic\_ios< char, \_Traits >::good(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

**4.300.5.59 read()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

## 4.300.5.60 readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called A here:

- if  $A == -1$ , sets eofbit and extracts no characters
- if  $A == 0$ , extracts no characters
- if  $A > 0$ , extracts  $\min(A, n)$

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file istream.tcc.

#### 4.300.5.61 register\_callback()

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

##### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

#### 4.300.5.62 seekg() [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

##### Returns

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.



**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**4.300.5.63 seekg()** [2/2]

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**4.300.5.64 setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**4.300.5.65 setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

**4.300.5.66 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::ws().

#### 4.300.5.67 str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istream< _CharT, _Traits, _Alloc >::str () const [inline]
```

Copying out the string buffer.

##### Returns

rdbuf() -> str()

Definition at line 519 of file sstream.

#### 4.300.5.68 str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istream< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

##### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls rdbuf() -> str(s).

Definition at line 529 of file sstream.

#### 4.300.5.69 sync()

```
template<typename _CharT, typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.

**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

**4.300.5.70 sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**4.300.5.71 tellg()**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**4.300.5.72 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.300.5.73 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.300.5.74 `unget()`**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void) [inherited]
```

Unextracting the previous character.

**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.

If `rdbuf()` is null or if `sungetc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 754 of file `istream.tcc`.

Referenced by `std::__detail::operator>>()`.

**4.300.5.75 `unsetf()`**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file ios\_base.h.

#### 4.300.5.76 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

##### Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file basic\_ios.h.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

#### 4.300.5.77 width() [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

##### Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _Outiter >::do_put()`.

#### 4.300.5.78 width() [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file `ios_base.h`.

**4.300.5.79 `xalloc()`**

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

**4.300.6 Member Data Documentation****4.300.6.1 `_M_gcount`**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, and `std::basic_istream< char >::unget()`.



#### 4.300.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.300.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

#### 4.300.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.300.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 4.300.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _Initer >::do_get()`, `std::num_put< _CharT, _Outiter >::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.300.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.300.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.300.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, and `std::noboolalpha()`.

#### 4.300.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.300.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::dec().

#### 4.300.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.300.6.13 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 4.300.6.14 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry() and std::basic\_ostream< \_CharT, \_Traits >::flush().

#### 4.300.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().

#### 4.300.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 402 of file ios\_base.h.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

#### 4.300.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

#### 4.300.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

#### 4.300.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow()`.

#### 4.300.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.300.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 4.300.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.300.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 4.300.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file `ios_base.h`.

Referenced by `std::right()`.

#### 4.300.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::hexfloat()`, and `std::scientific()`.

#### 4.300.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.

#### 4.300.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.300.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.300.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.300.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.300.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by `std::nounitbuf()`, and `std::unitbuf()`.

#### 4.300.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

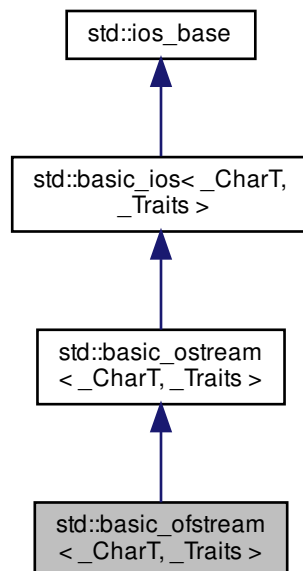
Referenced by `std::num_put<_CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

### 4.301 std::basic\_ofstream<\_CharT, \_Traits> Class Template Reference

Inheritance diagram for `std::basic_ofstream<_CharT, _Traits>`:





## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_filebuf](#)< char\_type, traits\_type > **\_\_filebuf\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
  - typedef [basic\\_ostream](#)< char\_type, traits\_type > **\_\_ostream\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef int io\_state **\_GLIBCXX\_DEPRECATED\_SUGGEST("std::iostate")**
  - typedef int open\_mode **\_GLIBCXX\_DEPRECATED\_SUGGEST("std::openmode")**
  - typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST("std::seekdir")**
  - typedef [std::streampos](#) **streampos** **\_GLIBCXX\_DEPRECATED\_SUGGEST("std::streampos")**
  - typedef [std::streamoff](#) **streamoff** **\_GLIBCXX\_DEPRECATED\_SUGGEST("std::streamoff")**
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) (event \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags **fmtflags**
  - typedef traits\_type::int\_type **int\_type**
  - typedef \_ios\_istate **iostate**
  - typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode **openmode**
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**

## Public Member Functions

- [basic\\_ofstream](#) ()
- [basic\\_ofstream](#) (const char \* \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- [basic\\_ofstream](#) (const [std::string](#) & \_\_s, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::out](#))
- **basic\_ofstream** (const [basic\\_ofstream](#) &)=delete
- **basic\_ofstream** ([basic\\_ofstream](#) && \_\_rhs)
- [~basic\\_ofstream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT >  
[basic\\_ostream](#)< \_CharT, \_Traits > & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- void [close](#) ()
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const

- `char_type fill () const`
  - `char_type fill (char_type __ch)`
  - `fmtflags flags () const`
  - `fmtflags flags (fmtflags __fmtfl)`
  - `__ostream_type & flush ()`
  - `locale getloc () const`
  - `bool good () const`
  - `locale imbue (const locale & __loc)`
  - `bool is_open ()`
  - `bool is_open () const`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `void open (const char * __s, ios_base::openmode __mode=ios_base::out)`
  - `void open (const std::string & __s, ios_base::openmode __mode=ios_base::out)`
  - `__ostream_type & operator<< (const void * __p)`
  - `__ostream_type & operator<< (__streambuf_type * __sb)`
  - `basic_ofstream & operator= (const basic_ofstream &)=delete`
  - `basic_ofstream & operator= (basic_ofstream && __rhs)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > * __sb)`
  - `__filebuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `void swap (basic_ofstream & __rhs)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > * __tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(*__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- `__ostream_type & operator<< (double __f)`
- `__ostream_type & operator<< (float __f)`
- `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type *__s, streamsize __n)`
- `__ostream_type & write (const char_type *__s, streamsize __n)`

- `operator bool () const`
- `bool operator! () const`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { **[\\_S\\_local\\_word\\_size](#)** }

### Protected Member Functions

- void **[\\_M\\_cache\\_locale](#)** (const [locale](#) & \_\_loc)
- void **[\\_M\\_call\\_callbacks](#)** ([event](#) \_\_ev) throw ()
- void **[\\_M\\_dispose\\_callbacks](#)** (void) throw ()
- [\\_Words](#) & **[\\_M\\_grow\\_words](#)** (int \_\_index, bool \_\_iword)
- void **[\\_M\\_init](#)** () throw ()
- template<typename [\\_ValueT](#) >  
[\\_\\_ostream\\_type](#) & **[\\_M\\_insert](#)** ([\\_ValueT](#) \_\_v)
- void **[\\_M\\_move](#)** ([ios\\_base](#) &) noexcept
- void **[\\_M\\_swap](#)** ([ios\\_base](#) & \_\_rhs) noexcept
- void **[init](#)** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void **[move](#)** ([basic\\_ios](#) & \_\_rhs)
- void **[move](#)** ([basic\\_ios](#) && \_\_rhs)
- void **[set\\_rdbuf](#)** ([basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > \* \_\_sb)
- void **[swap](#)** ([basic\\_ostream](#) & \_\_rhs)
- void **[swap](#)** ([basic\\_ios](#) & \_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf< _CharT, _Traits > * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream< _CharT, _Traits > * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 4.301.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ofstream< _CharT, _Traits >
```

Controlling output for files.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This class supports reading from named files, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_filebuf` is used, which this page refers to as `sb`.

Definition at line 753 of file `fstream`.

## 4.301.2 Member Typedef Documentation

4.301.2.1 `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 91 of file `basic_ios.h`.

#### 4.301.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

##### Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.301.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`

- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 341 of file `ios_base.h`.

#### 4.301.2.4 `iostate`

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 416 of file `ios_base.h`.

#### 4.301.2.5 `openmode`

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 447 of file `ios_base.h`.

#### 4.301.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.301.3 Member Enumeration Documentation

#### 4.301.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

### 4.301.4 Constructor & Destructor Documentation

#### 4.301.4.1 basic\_ofstream() [1/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream () [inline]
```

Default constructor.

Initializes `sb` using its default constructor, and passes `&sb` to the base class initializer. Does not open any files (you haven't given it a filename to open).

Definition at line 779 of file `fstream`.

#### 4.301.4.2 basic\_ofstream() [2/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
 const char * __s,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.



## Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | Null terminated string specifying the filename.  |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::out` is automatically included in `__mode`.

Definition at line 790 of file fstream.

## 4.301.4.3 basic\_ofstream() [3/3]

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Create an output file stream.

## Parameters

|                     |                                                  |
|---------------------|--------------------------------------------------|
| <code>__s</code>    | std::string specifying the filename.             |
| <code>__mode</code> | Open file in specified mode (see std::ios_base). |

`ios_base::out` is automatically included in `__mode`.

Definition at line 825 of file fstream.

## 4.301.4.4 ~basic\_ofstream()

```
template<typename _CharT , typename _Traits >
std::basic_ofstream< _CharT, _Traits >::~~basic_ofstream () [inline]
```

The destructor does nothing.

The file is closed by the filebuf object, not the formatting stream.

Definition at line 862 of file fstream.

## 4.301.5 Member Function Documentation

#### 4.301.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _Inlter>::do_get()`, `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_time()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_put<_CharT, _Outlter>::do_put()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::time_get<_CharT, _Inlter>::get()`, and `std::time_put<_CharT, _Outlter>::put()`.

#### 4.301.5.2 `_M_write()`

```
template<typename _CharT, typename _Traits>
void std::basic_ostream<_CharT, _Traits>::_M_write (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Core write functionality, without sentry.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 317 of file `ostream`.

#### 4.301.5.3 `bad()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.301.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

**4.301.5.5 close()**

```
template<typename _CharT , typename _Traits >
void std::basic_ofstream< _CharT, _Traits >::close () [inline]
```

Close the file.

Calls std::basic\_filebuf::close(). If that function fails, failbit is set in the stream's error state.

Definition at line 993 of file fstream.

**4.301.5.6 copyfmt()**

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the `pword` and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy `exceptions()`.

Definition at line 63 of file `basic_ios.tcc`.

**4.301.5.7 eof()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

**4.301.5.8 exceptions() [1/2]**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

**Returns**

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

**4.301.5.9 exceptions() [2/2]**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.301.5.10 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

## 4.301.5.11 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

## Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.301.5.12** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ' ) in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.301.5.13** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**4.301.5.14** `flags()` [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file `ios_base.h`.

## 4.301.5.15 flush()

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

## Returns

`*this`

If `rdbuf ( )` is a null pointer, changes nothing.

Otherwise, calls `rdbuf ( )->pubsync ( )`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

## 4.301.5.16 getloc()

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue (loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale ( )`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, and `std::ws()`.

#### 4.301.5.17 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.301.5.18 imbue()

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

##### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

##### Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 114 of file `basic_ios.tcc`.



## 4.301.5.19 init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::basic\_ios().

## 4.301.5.20 is\_open()

```
template<typename _CharT , typename _Traits >
bool std::basic_ofstream< _CharT, _Traits >::is_open () [inline]
```

Wrapper to test for an open file.

## Returns

```
rdbuf() -> is_open()
```


Definition at line 903 of file fstream.

## 4.301.5.21 iword()

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

## Parameters

|                                                                                                       |                       |
|-------------------------------------------------------------------------------------------------------|-----------------------|
|  <code>__ix</code> | Index into the array. |
|-------------------------------------------------------------------------------------------------------|-----------------------|

## Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

#### 4.301.5.22 `narrow()`

```
template<typename _CharT, typename _Traits>
char std::basic_ios<_CharT, _Traits>::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).narrow(c,default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file `basic_ios.h`.

#### 4.301.5.23 `open()` [1/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ofstream<_CharT, _Traits>::open (
 const char * __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

##### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 921 of file `fstream`.

#### 4.301.5.24 open() [2/2]

```
template<typename _CharT , typename _Traits >
void std::basic_ofstream< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode = ios_base::out) [inline]
```

Opens an external file.

##### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

Calls `std::basic_filebuf::open(s,mode|out)`. If that function fails, `failbit` is set in the stream's error state.

Definition at line 960 of file `fstream`.

#### 4.301.5.25 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file `basic_ios.h`.

#### 4.301.5.26 operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 125 of file `basic_ios.h`.

**4.301.5.27 operator<<()** [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `iomanip` header.

Definition at line 108 of file `ostream`.

**4.301.5.28 operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `iomanip` header.

Definition at line 117 of file `ostream`.

**4.301.5.29 operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "`std::cout << std::endl`". For more information, see the `iomanip` header.

Definition at line 127 of file `ostream`.

**4.301.5.30 operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 166 of file ostream.

## 4.301.5.31 operator&lt;&lt;() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 170 of file ostream.

## 4.301.5.32 operator&lt;&lt;() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                   |                                      |
|-----------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                  |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 174 of file `ostream`.

**4.301.5.33 operator<<() [7/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                    |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file `ostream.tcc`.

**4.301.5.34 operator<<() [8/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file ostream.

## 4.301.5.35 operator&lt;&lt;() [9/17]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 106 of file ostream.tcc.

## 4.301.5.36 operator&lt;&lt;() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                                    |                                      |
|----------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|----------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 192 of file `ostream`.

**4.301.5.37 operator<<() [11/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                                      |                                      |
|------------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>__n</code> | A variable of builtin integral type. |
|------------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file `ostream`.

**4.301.5.38 operator<<() [12/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.



## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

## 4.301.5.39 operator&lt;&lt;() [13/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|                  |                                            |
|------------------|--------------------------------------------|
| <code>__f</code> | A variable of builtin floating point type. |
|------------------|--------------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

## 4.301.5.40 operator&lt;&lt;() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ofstream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

**4.301.5.41 operator<<()** [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|          |                                            |
|----------|--------------------------------------------|
| ↵        | A variable of builtin floating point type. |
| ↵        |                                            |
| ↵        |                                            |
| ↵        |                                            |
| <i>f</i> |                                            |

**Returns**

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

**4.301.5.42 operator<<()** [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_p</code> | A variable of pointer type. |
|-----------------|-----------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

4.301.5.43 `operator<<()` [17/17]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

## Parameters

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set `failbit` in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets `failbit` in the error state

If the function inserts no characters, `failbit` is set.

Definition at line 120 of file `ostream.tcc`.

**4.301.5.44** `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

**4.301.5.45** `precision()` [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.

Definition at line 728 of file `ios_base.h`.

**4.301.5.46** `put()`

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert \_\_c.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

**4.301.5.47 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file ios\_base.h.

**4.301.5.48 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.301.5.49 `rdbuf()` [2/2]**

```
template<typename _CharT, typename _Traits >
__filebuf_type* std::basic_ofstream< _CharT, _Traits >::rdbuf () const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_filebuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 895 of file `fstream`.

**4.301.5.50 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unset()`.

**4.301.5.51 `register_callback()`**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

## 4.301.5.52 seekp() [1/2]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file `ostream.tcc`.

## 4.301.5.53 seekp() [2/2]

```
template<typename _CharT , typename _Traits >
basic_ofstream< _CharT, _Traits > & std::basic_ofstream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

## Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

#### 4.301.5.54 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

##### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 4.301.5.55 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

##### Parameters

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

##### Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.



Definition at line 693 of file ios\_base.h.

#### 4.301.5.56 setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

##### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See std::ios\_base::iostate for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by std::operator<<(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::ws().

#### 4.301.5.57 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 4.301.5.58 tellp()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**4.301.5.59 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.301.5.60 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

## 4.301.5.61 unsetf()

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

## 4.301.5.62 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.301.5.63** `width()` [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.301.5.64** `width()` [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file `ios_base.h`.

**4.301.5.65** `write()`

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

## Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

## 4.301.5.66 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

## Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `word` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `word` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `word` and `pword` arrays.

## 4.301.6 Member Data Documentation

#### 4.301.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outlter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.301.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

#### 4.301.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.301.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 4.301.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.301.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.301.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.301.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

#### 4.301.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`.

#### 4.301.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::dec().

#### 4.301.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.301.6.12 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 4.301.6.13 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::fail(), std::time\_get< \_CharT, \_InIter >::get(), and std::basic\_ostream< \_CharT, \_Traits >::sentry() and std::basic\_ostream< \_CharT, \_Traits >::flush().



## 4.301.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().

## 4.301.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of setf.

Definition at line 402 of file ios\_base.h.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

## 4.301.6.16 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

## 4.301.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

#### 4.301.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 4.301.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.301.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

#### 4.301.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.301.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 4.301.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file `ios_base.h`.

Referenced by `std::right()`.

#### 4.301.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::hexfloat()`, and `std::scientific()`.

#### 4.301.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::noshowbase()`, and `std::showbase()`.

**4.301.6.26 showpoint**

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.301.6.27 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.301.6.28 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

**4.301.6.29 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

## 4.301.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

## 4.301.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

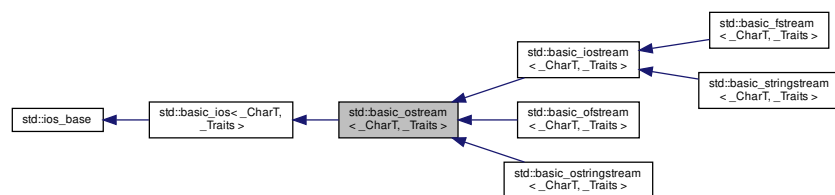
Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following file:

- [fstream](#)

## 4.302 std::basic\_ostream&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::basic\_ostream< \_CharT, \_Traits >:



## Classes

- class [sentry](#)

## Public Types

- typedef [ctype](#)< \_CharT > **\_\_ctype\_type**
  - typedef [basic\\_ios](#)< \_CharT, \_Traits > **\_\_ios\_type**
  - typedef [num\\_put](#)< \_CharT, [ostreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_put\_type**
  - typedef [basic\\_ostream](#)< \_CharT, \_Traits > **\_\_ostream\_type**
  - typedef [basic\\_streambuf](#)< \_CharT, \_Traits > **\_\_streambuf\_type**
  - typedef int io\_state **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::iostate")
  - typedef int open\_mode **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::openmode")
  - typedef int seek\_dir **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::seekdir")
  - typedef [std::streampos](#) **streampos** **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streampos")
  - typedef [std::streamoff](#) **streamoff** **\_GLIBCXX\_DEPRECATED\_SUGGEST**("std::streamoff")
  - typedef \_CharT **char\_type**
  - enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
  - typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) & \_\_b, int \_\_i)
  - typedef \_ios\_Fmtflags **fmtflags**
  - typedef \_Traits::int\_type **int\_type**
  - typedef \_ios\_iostate **iostate**
  - typedef \_Traits::off\_type **off\_type**
  - typedef \_ios\_Openmode **openmode**
  - typedef \_Traits::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef \_Traits **traits\_type**
- 
- typedef [num\\_get](#)< \_CharT, [istreambuf\\_iterator](#)< \_CharT, \_Traits > > **\_\_num\_get\_type**

## Public Member Functions

- [basic\\_ostream](#) ([\\_\\_streambuf\\_type](#) \* \_\_sb)
- virtual [~basic\\_ostream](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- template<typename \_ValueT >  
[basic\\_ostream](#)< \_CharT, \_Traits > & [\\_M\\_insert](#) (\_ValueT \_\_v)
- void [\\_M\\_setstate](#) ([iostate](#) \_\_state)
- bool [bad](#) () const
- void [clear](#) ([iostate](#) \_\_state=[goodbit](#))
- [basic\\_ios](#) & [copyfmt](#) (const [basic\\_ios](#) & \_\_rhs)
- bool [eof](#) () const
- [iostate](#) [exceptions](#) () const
- void [exceptions](#) ([iostate](#) \_\_except)
- bool [fail](#) () const
- [char\\_type](#) [fill](#) () const
- [char\\_type](#) [fill](#) ([char\\_type](#) \_\_ch)
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [\\_\\_ostream\\_type](#) & [flush](#) ()
- [locale](#) [getloc](#) () const

- `bool good () const`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf () const`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags setf (fmtflags __fmtfl)`
  - `fmtflags setf (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
- `__ostream_type & operator<< (unsigned long __n)`
- `__ostream_type & operator<< (bool __n)`
- `__ostream_type & operator<< (short __n)`
- `__ostream_type & operator<< (unsigned short __n)`
- `__ostream_type & operator<< (int __n)`
- `__ostream_type & operator<< (unsigned int __n)`
- `__ostream_type & operator<< (long long __n)`
- `__ostream_type & operator<< (unsigned long long __n)`

- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
- [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- [\\_\\_ostream\\_type](#) & [put](#) (char\_type \_\_c)
- void [\\_M\\_write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
- [\\_\\_ostream\\_type](#) & [write](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)

- [operator bool](#) () const
- bool [operator!](#) () const

### Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)



- static const [openmode in](#)
- static const [fmtflags internal](#)
- static const [fmtflags left](#)
- static const [fmtflags oct](#)
- static const [openmode out](#)
- static const [fmtflags right](#)
- static const [fmtflags scientific](#)
- static const [fmtflags showbase](#)
- static const [fmtflags showpoint](#)
- static const [fmtflags showpos](#)
- static const [fmtflags skipws](#)
- static const [openmode trunc](#)
- static const [fmtflags unitbuf](#)
- static const [fmtflags uppercase](#)

#### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

#### Protected Member Functions

- **basic\_ostream** ([basic\\_ostream](#)< \_CharT, \_Traits > &)
- **basic\_ostream** (const [basic\\_ostream](#) &)=delete
- **basic\_ostream** ([basic\\_ostream](#) &&\_\_rhs)
- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
    [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- [basic\\_ostream](#) & **operator=** (const [basic\\_ostream](#) &)=delete
- [basic\\_ostream](#) & **operator=** ([basic\\_ostream](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `const __ctype_type * _M_ctype`
- `iosstate _M_exception`
- `char_type _M_fill`
- `bool _M_fill_init`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [_S_local_word_size]`
- `const __num_get_type * _M_num_get`
- `const __num_put_type * _M_num_put`
- `streamsize _M_precision`
- `basic_streambuf<_CharT, _Traits> * _M_streambuf`
- `iosstate _M_streambuf_state`
- `basic_ostream<_CharT, _Traits> * _M_tie`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## Friends

- class `sentry`

## 4.302.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream<_CharT, _Traits>
```

Template class `basic_ostream`.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from `basic_streambuf` to do the actual output.

Definition at line 86 of file `iosfwd`.

## 4.302.2 Member Typedef Documentation

#### 4.302.2.1 \_\_num\_get\_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.




Definition at line 91 of file basic\_ios.h.

#### 4.302.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

##### Parameters

|                                                                                                      |                                                        |
|------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
|  <code>__e</code>   | One of the members of the event enum.                  |
|  <code>__b</code>   | Reference to the ios_base object.                      |
|  <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several ios\_base and basic\_ios functions, specifically imbue(), copyfmt(), and ~ios().

Definition at line 529 of file ios\_base.h.

#### 4.302.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex

- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 341 of file ios\_base.h.

#### 4.302.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 416 of file ios\_base.h.

#### 4.302.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.302.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.302.3 Member Enumeration Documentation

#### 4.302.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

#### 4.302.4 Constructor & Destructor Documentation

##### 4.302.4.1 `basic_ostream()`

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::basic_ostream (
 __streambuf_type * __sb) [inline], [explicit]
```

Base constructor.

This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

Definition at line 84 of file ostream.

##### 4.302.4.2 `~basic_ostream()`

```
template<typename _CharT, typename _Traits>
virtual std::basic_ostream< _CharT, _Traits >::~~basic_ostream () [inline], [virtual]
```

Base destructor.

This does very little apart from providing a virtual base dtor.

Definition at line 93 of file ostream.

#### 4.302.5 Member Function Documentation

##### 4.302.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

##### 4.302.5.2 `_M_write()`

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
 const char_type * __s,
 streamsize __n) [inline]
```

Core write functionality, without sentry.

**Parameters**

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>_↵<br/>_s</code> | The array to insert.                    |
| <code>_↵<br/>_n</code> | Maximum number of characters to insert. |

Definition at line 317 of file ostream.

**4.302.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.302.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >↵::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 4.302.5.5 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 4.302.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 4.302.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().



## 4.302.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.302.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

**4.302.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.302.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.302.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**4.302.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

**4.302.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
```

Synchronizing the stream buffer.

**Returns**

`*this`

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

#### 4.302.5.15 getloc()

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, and `std::ws()`.

#### 4.302.5.16 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::basic_istream<_CharT, _Traits>::sentry::sentry()`.

#### 4.302.5.17 imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios<_CharT, _Traits>::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

##### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**4.302.5.18 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**4.302.5.19 iword()**

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file ios\_base.h.

#### 4.302.5.20 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>↔

Definition at line 430 of file basic\_ios.h.

#### 4.302.5.21 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.

## 4.302.5.22 operator!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

## 4.302.5.23 operator&lt;&lt;() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

## 4.302.5.24 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ios_type &(*) (__ios_type &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

## 4.302.5.25 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.


Definition at line 127 of file `ostream`.

## 4.302.5.26 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                   |                                      |
|-----------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                  |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**4.302.5.27 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                    |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.302.5.28 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline]
```

Integer arithmetic inserters.



## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↵</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 4.302.5.29 operator&lt;&lt;() [7/17]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n)
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>_↵</code>  | A variable of builtin integral type. |
| <code>__n</code> |                                      |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 4.302.5.30 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>__</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**4.302.5.31 operator<<() [9/17]**

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n)
```

Integer arithmetic inserters.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>__</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**4.302.5.32 operator<<() [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 4.302.5.33 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 4.302.5.34 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline]
```

Integer arithmetic inserters.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**4.302.5.35 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline]
```

Floating point arithmetic inserters.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**4.302.5.36 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 4.302.5.37 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long double __f) [inline]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 4.302.5.38 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 const void * __p) [inline]
```

Pointer arithmetic inserters.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.302.5.39 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb)
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

## 4.302.5.40 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), and std::operator<<().

## 4.302.5.41 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

Definition at line 728 of file ios\_base.h.

## 4.302.5.42 put()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c)
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

**Returns**

\*this

Tries to insert \_\_c.

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 149 of file ostream.tcc.

**4.302.5.43 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file ios\_base.h.

**4.302.5.44 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::rdbuf () const [inline],
[inherited]
```

Accessing the underlying buffer.



**Returns**

The current stream buffer.

This does not change the state of the stream.

Definition at line 321 of file basic\_ios.h.

Referenced by std::ws().

**4.302.5.45 rdbuf()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
 basic_streambuf< _CharT, _Traits > * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file basic\_ios.tcc.

**4.302.5.46 rdstate()**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

**4.302.5.47 register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.302.5.48 seekp()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos)
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets `failbit`.

Definition at line 258 of file `ostream.tcc`.

## 4.302.5.49 seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 off_type __off,
 ios_base::seekdir __dir)
```

Changing the current write position.

## Parameters

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

## Returns

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

## 4.302.5.50 setf() [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

## Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**4.302.5.51** `setf()` [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

**Returns**

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

**4.302.5.52** `setstate()`

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

**4.302.5.53** `sync_with_stdio()`

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

## 4.302.5.54 tellp()

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

## 4.302.5.55 tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

## Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

## 4.302.5.56 tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.302.5.57 unsetf()**

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.302.5.58 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
| <code>__C</code> |                         |

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

#### 4.302.5.59 width() [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

#### 4.302.5.60 width() [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

Returns

The previous value of `width()`.

Definition at line 751 of file `ios_base.h`.

#### 4.302.5.61 write()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n)
```

Character string insertion.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

##### Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, `badbit` will be set in the stream's error state)

##### Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file `ostream.tcc`.

#### 4.302.5.62 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.



#### 4.302.6 Member Data Documentation

##### 4.302.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

##### 4.302.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

##### 4.302.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

##### 4.302.6.4 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 4.302.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.302.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file `ios_base.h`.

#### 4.302.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary](https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary).

Definition at line 458 of file `ios_base.h`.

#### 4.302.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_↵put()`, and `std::noboolalpha()`.

#### 4.302.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.302.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::dec().

#### 4.302.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.302.6.12 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 4.302.6.13 failbit

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry()`.

#### 4.302.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::hexfloat()`.

#### 4.302.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.302.6.16 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsome()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std::basic_istream< char >::unget()`.

#### 4.302.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_Inlter >::do\_get(), std::num\_put< \_CharT, \_Outlter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

#### 4.302.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for ifstream and fstream.

Definition at line 461 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

#### 4.302.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 358 of file ios\_base.h.

Referenced by std::internal().

#### 4.302.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_Outlter >::do\_put(), and std::left().

#### 4.302.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file ios\_base.h.

Referenced by std::oct(), and std::basic\_ostream< char >::operator<<().

#### 4.302.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for ofstream and fstream.

Definition at line 464 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::pbackfail(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos().

#### 4.302.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

#### 4.302.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

#### 4.302.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_Outlter>::do\_put(), std::noshowbase(), and std::showbase().

#### 4.302.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.302.6.27 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.302.6.28 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.302.6.29 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 467 of file `ios_base.h`.

#### 4.302.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file `ios_base.h`.

Referenced by `std::nunitbuf()`, and `std::unitbuf()`.

#### 4.302.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

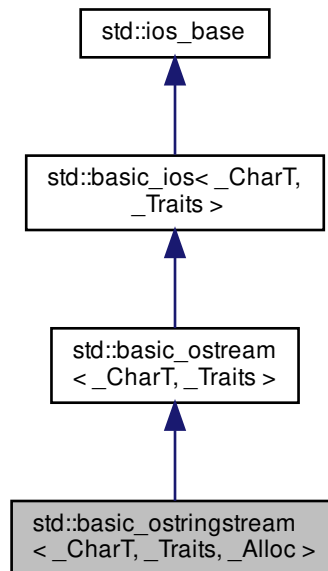
The documentation for this class was generated from the following files:

- [iosfwd](#)
- [ostream](#)
- [ostream.tcc](#)



## 4.303 std::basic\_ostringstream&lt; \_CharT, \_Traits, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >:



## Public Types

- typedef `ctype< _CharT > __ctype_type`
- typedef `basic_ios< _CharT, _Traits > __ios_type`
- typedef `num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type`
- typedef `basic_ostream< char_type, traits_type > __ostream_type`
- typedef `basic_streambuf< _CharT, _Traits > __streambuf_type`
- typedef `basic_string< _CharT, _Traits, _Alloc > __string_type`
- typedef `basic_stringbuf< _CharT, _Traits, _Alloc > __stringbuf_type`
- typedef `int io_state _GLIBCXX_DEPRECATED_SUGGEST("std::iostate")`
- typedef `int open_mode _GLIBCXX_DEPRECATED_SUGGEST("std::openmode")`
- typedef `int seek_dir _GLIBCXX_DEPRECATED_SUGGEST("std::seekdir")`
- typedef `std::streampos streampos _GLIBCXX_DEPRECATED_SUGGEST("std::streampos")`
- typedef `std::streamoff streamoff _GLIBCXX_DEPRECATED_SUGGEST("std::streamoff")`
- typedef `_Alloc allocator_type`
- typedef `_CharT char_type`
- enum `event { erase_event, imbue_event, copyfmt_event }`
- typedef `void(* event_callback) (event __e, ios_base & __b, int __i)`
- typedef `_ios_Fmtflags fmtflags`
- typedef `traits_type::int_type int_type`
- typedef `_ios_losestate iostate`

- typedef traits\_type::off\_type **off\_type**
  - typedef \_ios\_Openmode **openmode**
  - typedef traits\_type::pos\_type **pos\_type**
  - typedef \_ios\_Seekdir **seekdir**
  - typedef \_Traits **traits\_type**
- 
- typedef num\_get< \_CharT, istreambuf\_iterator< \_CharT, \_Traits > > **\_\_num\_get\_type**

#### Public Member Functions

- **basic\_ostringstream** ()
- **basic\_ostringstream** (ios\_base::openmode \_\_mode)
- **basic\_ostringstream** (const \_\_string\_type &\_\_str, ios\_base::openmode \_\_mode=ios\_base::out)
- **basic\_ostringstream** (const **basic\_ostringstream** &)=delete
- **basic\_ostringstream** (**basic\_ostringstream** &&\_\_rhs)
- **~basic\_ostringstream** ()
- const locale & **\_M\_getloc** () const
- template<typename \_ValueT >  
**basic\_ostream**< \_CharT, \_Traits > & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_setstate** (iostate \_\_state)
- bool **bad** () const
- void **clear** (iostate \_\_state=goodbit)
- **basic\_ios** & **copyfmt** (const **basic\_ios** &\_\_rhs)
- bool **eof** () const
- iostate **exceptions** () const
- void **exceptions** (iostate \_\_except)
- bool **fail** () const
- char\_type **fill** () const
- char\_type **fill** (char\_type \_\_ch)
- **fmtflags** **flags** () const
- **fmtflags** **flags** (**fmtflags** \_\_fmtfl)
- **\_\_ostream\_type** & **flush** ()
- locale **getloc** () const
- bool **good** () const
- locale **imbue** (const locale &\_\_loc)
- long & **iword** (int \_\_ix)
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- **\_\_ostream\_type** & **operator<<** (const void \*\_\_p)
- **\_\_ostream\_type** & **operator<<** (**\_\_streambuf\_type** \*\_\_sb)
- **basic\_ostringstream** & **operator=** (const **basic\_ostringstream** &)=delete
- **basic\_ostringstream** & **operator=** (**basic\_ostringstream** &&\_\_rhs)
- **streamsize** **precision** () const
- **streamsize** **precision** (**streamsize** \_\_prec)
- void \*& **pword** (int \_\_ix)
- **basic\_streambuf**< \_CharT, \_Traits > \* **rdbuf** (**basic\_streambuf**< \_CharT, \_Traits > \*\_\_sb)
- **\_\_stringbuf\_type** \* **rdbuf** () const
- iostate **rdstate** () const

- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([pos\\_type](#))
  - [\\_\\_ostream\\_type](#) & [seekp](#) ([off\\_type](#), [ios\\_base::seekdir](#))
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
  - [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
  - void [setstate](#) ([iostate](#) \_\_state)
  - [\\_\\_string\\_type](#) [str](#) () const
  - void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
  - void [swap](#) ([basic\\_ostringstream](#) &\_\_rhs)
  - [pos\\_type](#) [tellp](#) ()
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) () const
  - [basic\\_ostream](#)< \_CharT, \_Traits > \* [tie](#) ([basic\\_ostream](#)< \_CharT, \_Traits > \* \_\_tiestr)
  - void [unsetf](#) ([fmtflags](#) \_\_mask)
  - [char\\_type](#) [widen](#) ([char](#) \_\_c) const
  - [streamsize](#) [width](#) () const
  - [streamsize](#) [width](#) ([streamsize](#) \_\_wide)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ostream\\_type](#) &(\*\_\_pf)([\\_\\_ostream\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([\\_\\_ios\\_type](#) &(\*\_\_pf)([\\_\\_ios\\_type](#) &))
  - [\\_\\_ostream\\_type](#) & [operator<<](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the [sentry](#) documentation for more.

If the `sentry` status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- [\\_\\_ostream\\_type](#) & [operator<<](#) (long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (bool \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned short \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned int \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long long \_\_n)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (unsigned long long \_\_n)
- 
- [\\_\\_ostream\\_type](#) & [operator<<](#) (double \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (float \_\_f)
  - [\\_\\_ostream\\_type](#) & [operator<<](#) (long double \_\_f)

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
  - `void _M_write (const char_type *__s, streamsize __n)`
  - `__ostream_type & write (const char_type *__s, streamsize __n)`
- 
- `operator bool () const`
  - `bool operator! () const`

### Static Public Member Functions

- `static bool sync_with_stdio (bool __sync=true)`
- `static int xalloc () throw ()`

### Static Public Attributes

- `static const fmtflags adjustfield`
- `static const openmode app`
- `static const openmode ate`
- `static const iostate badbit`
- `static const fmtflags basefield`
- `static const seekdir beg`
- `static const openmode binary`
- `static const fmtflags boolalpha`
- `static const seekdir cur`
- `static const fmtflags dec`
- `static const seekdir end`
- `static const iostate eofbit`
- `static const iostate failbit`
- `static const fmtflags fixed`
- `static const fmtflags floatfield`
- `static const iostate goodbit`
- `static const fmtflags hex`
- `static const openmode in`
- `static const fmtflags internal`
- `static const fmtflags left`
- `static const fmtflags oct`
- `static const openmode out`
- `static const fmtflags right`
- `static const fmtflags scientific`
- `static const fmtflags showbase`
- `static const fmtflags showpoint`
- `static const fmtflags showpos`
- `static const fmtflags skipws`
- `static const openmode trunc`
- `static const fmtflags unitbuf`
- `static const fmtflags uppercase`

## Protected Types

- enum { **\_S\_local\_word\_size** }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) & \_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) & \_\_rhs)
- void **move** ([basic\\_ios](#) && \_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) & \_\_rhs)
- void **swap** ([basic\\_ios](#) & \_\_rhs) noexcept

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.303.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_ostringstream< _CharT, _Traits, _Alloc >
```

Controlling output for std::string.

### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_ostream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 104 of file `iosfwd`.

### 4.303.2 Member Typedef Documentation

#### 4.303.2.1 `__num_get_type`

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_get_type [inherited]
```

These are non-standard types.

Definition at line 91 of file `basic_ios.h`.

#### 4.303.2.2 `event_callback`

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

### Parameters

|                       |                                                        |
|-----------------------|--------------------------------------------------------|
| <code>↔<br/>_e</code> | One of the members of the event enum.                  |
| <code>↔<br/>_b</code> | Reference to the <code>ios_base</code> object.         |
| <code>↔<br/>_i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.303.2.3 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

*\_Ios\_Fmtflags* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *fmtflags* are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Definition at line 341 of file *ios\_base.h*.

#### 4.303.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 416 of file *ios\_base.h*.

#### 4.303.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.303.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.303.3 Member Enumeration Documentation

#### 4.303.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.



## 4.303.4 Constructor &amp; Destructor Documentation

## 4.303.4.1 basic\_ostringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 583 of file `sstream`.

## 4.303.4.2 basic\_ostringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

`ios_base::out` is automatically included in `mode`.

Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 600 of file `sstream`.

## 4.303.4.3 basic\_ostringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.

## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

`ios_base::out` is automatically included in *mode*.

Initializes `sb` using *str* and *mode*|out, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 618 of file `sstream`.

4.303.4.4 `~basic_ostringstream()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~basic_ostringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 629 of file `sstream`.

## 4.303.5 Member Function Documentation

4.303.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

## Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

4.303.5.2 `_M_write()`

```
template<typename _CharT, typename _Traits>
void std::basic_ostream< _CharT, _Traits >::_M_write (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Core write functionality, without sentry.

**Parameters**

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>_↵<br/>_s</code> | The array to insert.                    |
| <code>_↵<br/>_n</code> | Maximum number of characters to insert. |

Definition at line 317 of file ostream.

**4.303.5.3 bad()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const [inline], [inherited]
```

Fast error checking.

**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.303.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

#### 4.303.5.5 copyfmt()

```
template<typename _CharT, typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of \_\_rhs into this.

##### Parameters

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

##### Returns

Reference to this object.

All fields of \_\_rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase\_event. After copying, each (new) callback is invoked with copyfmt\_event. The final step is to copy exceptions().

Definition at line 63 of file basic\_ios.tcc.

#### 4.303.5.6 eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

Definition at line 190 of file basic\_ios.h.

#### 4.303.5.7 exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

Definition at line 222 of file basic\_ios.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt().

## 4.303.5.8 exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.303.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other `iostate` flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits >::operator bool()`, and `std::basic_ios<char, _Traits >::operator!()`.

**4.303.5.10** `fill()` [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

**Returns**

The current fill character.

It defaults to a space ( ' ') in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.303.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.303.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::basic\_ostream< char >::operator<<(), std::operator<<(), std::\_\_detail::operator>>(), and std::operator>>().

**4.303.5.13 flags()** [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file ios\_base.h.

**4.303.5.14 flush()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf() -> pubsync()`, and if that returns -1, sets badbit.

Definition at line 211 of file ostream.tcc.

#### 4.303.5.15 `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _OutIter >::do_put()`, and `std::ws()`.

#### 4.303.5.16 `good()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if no error flags are set.

A wrapper around `rdstate`.

Definition at line 180 of file `basic_ios.h`.

Referenced by `std::__detail::operator>>()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::basic_istream< _CharT, _Traits >::sentry::sentry()`.

#### 4.303.5.17 `imbue()`

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

##### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|



**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**4.303.5.18 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
 basic_streambuf< _CharT, _Traits > * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios< char, _Traits >::basic_ios()`.

**4.303.5.19 iword()**

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file ios\_base.h.

#### 4.303.5.20 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file basic\_ios.h.

#### 4.303.5.21 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.

## 4.303.5.22 operator!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

## 4.303.5.23 operator&lt;&lt;() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

## 4.303.5.24 operator&lt;&lt;() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

## 4.303.5.25 operator&lt;&lt;() [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.


Definition at line 127 of file `ostream`.

## 4.303.5.26 operator&lt;&lt;() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                   |                                      |
|-----------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                  |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**4.303.5.27 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                    |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.303.5.28 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 4.303.5.29 operator&lt;&lt;() [7/17]

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 4.303.5.30 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**4.303.5.31 `operator<<()` [9/17]**

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                    |                                      |
|------------------------------------|--------------------------------------|
| <code>_↵</code><br><code>_n</code> | A variable of builtin integral type. |
|------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**4.303.5.32 `operator<<()` [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 4.303.5.33 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 4.303.5.34 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**4.303.5.35 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**4.303.5.36 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.



## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 4.303.5.37 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 4.303.5.38 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.303.5.39 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

## 4.303.5.40 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

**Returns**

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::copyfmt(), and std::operator<<().

## 4.303.5.41 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of precision().

Definition at line 728 of file ios\_base.h.

## 4.303.5.42 put()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to insert. |
|------------------|--------------------------|

\*this

This function is not overloaded on signed char and unsigned char.

|   |                       |
|---|-----------------------|
| ← | Index into the array. |
|---|-----------------------|

|  |  |
|--|--|
|  |  |
|--|--|

A reference to a void\* associated with the index.

## Parameters

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

## Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

4.303.5.45 `rdbuf()` [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type* std::basic_ostringstream<_CharT, _Traits, _Alloc>::rdbuf () const [inline]
```

Accessing the underlying buffer.

## Returns

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 669 of file `sstream`.

4.303.5.46 `rdstate()`

```
template<typename _CharT, typename _Traits>
iosstate std::basic_ios<_CharT, _Traits>::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

## Returns

A bit pattern (well, isn't everything?)

See `std::ios_base::iosstate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::eof()`, `std::basic_ios<char, _Traits>::fail()`, `std::basic_ios<char, _Traits>::good()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_ios<char, _Traits>::setstate()`, and `std::basic_istream<char>::unset()`.

4.303.5.47 `register_callback()`

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.303.5.48 seekp()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

**4.303.5.49 seekp()** [2/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets `failbit`.

Definition at line 290 of file `ostream.tcc`.

#### 4.303.5.50 `setf()` [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

##### Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

##### Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

#### 4.303.5.51 `setf()` [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

##### Parameters

|                      |                                         |
|----------------------|-----------------------------------------|
| <code>__fmtfl</code> | Additional flags to set.                |
| <code>__mask</code>  | The flags mask for <code>fmtfl</code> . |

##### Returns

The previous format control flags.

This function clears `mask` in the format flags, then sets `fmtfl` & `mask`. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file ios\_base.h.

#### 4.303.5.52 setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

##### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file basic\_ios.h.

Referenced by `std::operator<<()`, `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`, and `std::ws()`.

#### 4.303.5.53 str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str () const [inline]
```

Copying out the string buffer.

##### Returns

```
rdbuf()->str()
```

Definition at line 677 of file sstream.

Referenced by `std::__detail::operator<<()`.

#### 4.303.5.54 str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.



## Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <code>_↵</code> | The string to use as a new sequence. |
| <code>_s</code> |                                      |

Calls `rdbuf() -> str(s)`.

Definition at line 687 of file `sstream`.

## 4.303.5.55 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

## Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

## Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

## 4.303.5.56 tellp()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
[inherited]
```

Getting the current write position.

## Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf() -> pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**4.303.5.57** `tie()` [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.303.5.58** `tie()` [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios<_CharT, _Traits >::tie (
 basic_ostream<_CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

**Returns**

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see `tie()` for more.

Definition at line 307 of file `basic_ios.h`.

**4.303.5.59** `unsetf()`

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

## Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

## 4.303.5.60 widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

## Returns

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

## 4.303.5.61 width() [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

## Returns

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.303.5.62 width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of width().

Definition at line 751 of file ios\_base.h.

**4.303.5.63 write()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

**Returns**

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 4.303.5.64 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

### 4.303.6 Member Data Documentation

#### 4.303.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.303.6.2 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

#### 4.303.6.3 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

#### 4.303.6.4 badbit

```
const ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::fail()`, and `std::operator<<()`.

#### 4.303.6.5 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of `dec|oct|hex`. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.303.6.6 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file `ios_base.h`.

#### 4.303.6.7 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file `ios_base.h`.

#### 4.303.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_←put()`, and `std::noboolalpha()`.

#### 4.303.6.9 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.303.6.10 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file `ios_base.h`.

Referenced by `std::dec()`.

#### 4.303.6.11 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`.

#### 4.303.6.12 eofbit

```
const ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_date()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_time()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::eof()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::seekg()`, `std::basic_istream<char>::unget()`, and `std::ws()`.

#### 4.303.6.13 failbit

```
const ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ios<char, _Traits>::fail()`, `std::time_get<_CharT, _InIter>::get()`, and `std::basic_ostream<_CharT, _Traits>::sentry()`.

#### 4.303.6.14 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::hexfloat()`.

#### 4.303.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.



## 4.303.6.16 goodbit

```
const ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ostream< char >::flush(), std::basic\_istream< char >::get(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::getline(), std::basic\_istream< char >::ignore(), std::basic\_ostream< char >::operator<<(), std::basic\_istream< char >::operator>>(), std::operator>>(), std::basic\_istream< char >::peek(), std::basic\_ostream< char >::put(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::read(), std::basic\_istream< char >::readsome(), std::basic\_istream< char >::seekg(), std::basic\_ostream< char >::seekp(), std::basic\_istream< \_CharT, \_Traits >::sentry::sentry(), std::basic\_istream< char >::sync(), and std::basic\_istream< char >::unget().

## 4.303.6.17 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

## 4.303.6.18 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for ifstream and fstream.

Definition at line 461 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

#### 4.303.6.19 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.303.6.20 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, and `std::left()`.

#### 4.303.6.21 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.303.6.22 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::pbackfail()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`.

#### 4.303.6.23 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

#### 4.303.6.24 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

#### 4.303.6.25 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter>::do\_put(), std::noshowbase(), and std::showbase().

#### 4.303.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

**4.303.6.27 showpos**

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

**4.303.6.28 skipws**

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

**4.303.6.29 trunc**

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

**4.303.6.30 unitbuf**

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

## 4.303.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::nouppercase(), and std::uppercase().

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

## 4.304 std::basic\_regex&lt; \_Ch\_type, \_Rx\_traits &gt; Class Template Reference

## Public Types

- typedef [regex\\_constants::syntax\\_option\\_type](#) **flag\_type**
- typedef traits\_type::locale\_type **locale\_type**
- typedef traits\_type::string\_type **string\_type**
- typedef \_Rx\_traits **traits\_type**
- typedef \_Ch\_type **value\_type**

## Public Member Functions

- [basic\\_regex](#) ()
- [basic\\_regex](#) (const \_Ch\_type \* \_\_p, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const \_Ch\_type \* \_\_p, std::size\_t \_\_len, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) (const [basic\\_regex](#) & \_\_rhs)=default
- [basic\\_regex](#) ([basic\\_regex](#) && \_\_rhs) noexcept=default
- template<typename \_Ch\_traits, typename \_Ch\_alloc >  
  [basic\\_regex](#) (const [std::basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > & \_\_s, [flag\\_type](#) \_\_f=ECMAScript)
- template<typename \_FwdIter >  
  [basic\\_regex](#) (\_FwdIter \_\_first, \_FwdIter \_\_last, [flag\\_type](#) \_\_f=ECMAScript)
- [basic\\_regex](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_f=ECMAScript)
- ~[basic\\_regex](#) ()
- [basic\\_regex](#) & [assign](#) (const [basic\\_regex](#) & \_\_rhs)
- [basic\\_regex](#) & [assign](#) ([basic\\_regex](#) && \_\_rhs) noexcept
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \* \_\_p, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) (const \_Ch\_type \* \_\_p, size\_t \_\_len, [flag\\_type](#) \_\_flags=ECMAScript)
- template<typename \_Ch\_traits, typename \_Alloc >  
  [basic\\_regex](#) & [assign](#) (const [basic\\_string](#)< \_Ch\_type, \_Ch\_traits, \_Alloc > & \_\_s, [flag\\_type](#) \_\_flags=ECMAScript)
- template<typename \_InputIterator >  
  [basic\\_regex](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [flag\\_type](#) \_\_flags=ECMAScript)
- [basic\\_regex](#) & [assign](#) ([initializer\\_list](#)< \_Ch\_type > \_\_l, [flag\\_type](#) \_\_flags=ECMAScript)
- [flag\\_type](#) [flags](#) () const
- locale\_type [getloc](#) () const

- locale\_type imbue (locale\_type \_\_loc)
- unsigned int mark\_count () const
- basic\_regex & operator= (const basic\_regex &\_\_rhs)
- basic\_regex & operator= (basic\_regex &&\_\_rhs) noexcept
- basic\_regex & operator= (const \_Ch\_type \*\_\_p)
- basic\_regex & operator= (initializer\_list< \_Ch\_type > \_\_l)
- template<typename \_Ch\_traits , typename \_Alloc >  
basic\_regex & operator= (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Alloc > &\_\_s)
- void swap (basic\_regex &\_\_rhs)

## Static Public Attributes

### Constants

*std [28.8.1](1)*

- static constexpr flag\_type icase
- static constexpr flag\_type nosubs
- static constexpr flag\_type optimize
- static constexpr flag\_type collate
- static constexpr flag\_type ECMAScript
- static constexpr flag\_type basic
- static constexpr flag\_type extended
- static constexpr flag\_type awk
- static constexpr flag\_type grep
- static constexpr flag\_type egrep

## Friends

- template<typename \_Bp , typename \_Ap , typename \_Cp , typename \_Rp , \_\_detail::\_RegexExecutorPolicy , bool >  
bool \_\_detail::\_regex\_algo\_impl (\_Bp, \_Bp, match\_results< \_Bp, \_Ap > &, const basic\_regex< \_Cp, \_Rp > &, regex\_constants::match\_flag\_type)
- template<typename , typename , typename , bool >  
class \_\_detail::\_Executor

## Related Functions

(Note that these are not member functions.)

- template<typename \_Ch\_type , typename \_Rx\_traits >  
void swap (basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_lhs, basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_rhs)

### 4.304.1 Detailed Description

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex< _Ch_type, _Rx_traits >
```

Objects of specializations of this class represent regular expressions constructed from sequences of character type `_Ch_type`.

Storage for the regular expression is allocated and deallocated as necessary by the member functions of this class.

Definition at line 36 of file `regex.h`.

## 4.304.2 Constructor &amp; Destructor Documentation

## 4.304.2.1 basic\_regex() [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex () [inline]
```

Constructs a basic regular expression that does not match any character sequence.

Definition at line 423 of file regex.h.



Referenced by std::basic\_regex< \_Ch\_type, \_Rx\_traits >::assign().

## 4.304.2.2 basic\_regex() [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const _Ch_type * __p,
 flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the sequence [\_\_p, \_\_p + char\_traits<\_Ch\_type>::length(\_\_p)) interpreted according to the flags in \_\_f.

## Parameters

|                                                                                            |                                                                                             |
|--------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <br>__p | A pointer to the start of a C-style null-terminated string containing a regular expression. |
| <br>__f | Flags indicating the syntax rules and options.                                              |

## Exceptions

|                    |                                           |
|--------------------|-------------------------------------------|
| <i>regex_error</i> | if __p is not a valid regular expression. |
|--------------------|-------------------------------------------|

Definition at line 439 of file regex.h.

## 4.304.2.3 basic\_regex() [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const _Ch_type * __p,
```

```
std::size_t __len,
flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in f.



## Parameters

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>__p</code>   | A pointer to the start of a string containing a regular expression. |
| <code>__len</code> | The length of the string containing the regular expression.         |
| <code>__f</code>   | Flags indicating the syntax rules and options.                      |

## Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__p</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 455 of file regex.h.

## 4.304.2.4 basic\_regex() [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const basic_regex< _Ch_type, _Rx_traits > & __rhs) [default]
```

Copy-constructs a basic regular expression.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

## 4.304.2.5 basic\_regex() [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 basic_regex< _Ch_type, _Rx_traits > && __rhs) [default], [noexcept]
```

Move-constructs a basic regular expression.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__rhs</code> | A regex object. |
|--------------------|-----------------|

## 4.304.2.6 basic\_regex() [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits , typename _Ch_alloc >
```

```
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
 flag_type __f = ECMAScript) [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

#### Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__s</code> | A string containing a regular expression.      |
| <code>__f</code> | Flags indicating the syntax rules and options. |

#### Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 485 of file `regex.h`.

#### 4.304.2.7 `basic_regex()` [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter >
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
 _FwdIter __first,
 _FwdIter __last,
 flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from the range `[first, last)` interpreted according to the flags in `f`.

#### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__f</code>     | The format flags of the regular expression.                 |

#### Exceptions

|                          |                                                                      |
|--------------------------|----------------------------------------------------------------------|
| <code>regex_error</code> | if <code>[__first, __last)</code> is not a valid regular expression. |
|--------------------------|----------------------------------------------------------------------|

Definition at line 505 of file `regex.h`.

## 4.304.2.8 basic\_regex() [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex<_Ch_type, _Rx_traits >::basic_regex (
 initializer_list<_Ch_type > __l,
 flag_type __f = ECMAScript) [inline]
```

Constructs a basic regular expression from an initializer list.

## Parameters

|                                |                                             |
|--------------------------------|---------------------------------------------|
| ↩<br>_↩<br>↩<br>_↩<br><i>l</i> | The initializer list.                       |
| ↩<br>_↩<br>↩<br>_↩<br><i>f</i> | The format flags of the regular expression. |

## Exceptions

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <i>regex_error</i> | if <code>__l</code> is not a valid regular expression. |
|--------------------|--------------------------------------------------------|

Definition at line 518 of file regex.h.

## 4.304.2.9 ~basic\_regex()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex<_Ch_type, _Rx_traits >::~~basic_regex () [inline]
```

Destroys a basic regular expression.

Definition at line 525 of file regex.h.

## 4.304.3 Member Function Documentation

## 4.304.3.1 assign() [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
 const basic_regex<_Ch_type, _Rx_traits > & __rhs) [inline]
```

the real assignment operator.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Definition at line 583 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`, and `std::basic_regex<_Ch_type, _Rx_traits >::operator=()`.

**4.304.3.2 assign()** [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
 basic_regex<_Ch_type, _Rx_traits > && __rhs) [inline], [noexcept]
```

The move-assignment operator.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | Another regular expression object. |
|--------------------|------------------------------------|

Definition at line 596 of file `regex.h`.

References `std::move()`, and `std::basic_regex<_Ch_type, _Rx_traits >::swap()`.

**4.304.3.3 assign()** [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
 const _Ch_type * __p,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

**Parameters**

|                      |                                                                                        |
|----------------------|----------------------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style null-terminated string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                                                                   |

## Exceptions

|                    |                                                                                                                                                                                                      |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if <code>__p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 617 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

## 4.304.3.4 assign() [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
 const _Ch_type * __p,
 size_t __len,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

## Parameters

|                      |                                                                        |
|----------------------|------------------------------------------------------------------------|
| <code>__p</code>     | A pointer to a C-style string containing a regular expression pattern. |
| <code>__len</code>   | The length of the regular expression pattern string.                   |
| <code>__flags</code> | Syntax option flags.                                                   |

## Exceptions

|                    |                                                                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>regex_error</i> | if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 636 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

## 4.304.3.5 assign() [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits , typename _Alloc >
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::assign (
 const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

## Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__s</code>     | A string containing a regular expression pattern. |
| <code>__flags</code> | Syntax option flags.                              |

## Exceptions

|                          |                                                                                                                                                                                                      |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__s</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, <code>*this</code> remains unchanged. |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 652 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, `std::basic_regex<_Ch_type, _Rx_traits>::basic_regex()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::size()`.

4.304.3.6 `assign()` [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _InputIterator >
basic_regex& std::basic_regex<_Ch_type, _Rx_traits>::assign (
 _InputIterator __first,
 _InputIterator __last,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | The start of a range containing a valid regular expression. |
| <code>__last</code>  | The end of a range containing a valid regular expression.   |
| <code>__flags</code> | Syntax option flags.                                        |

## Exceptions

|                          |                                                                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>p</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged. |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 674 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits>::assign()`.

## 4.304.3.7 assign() [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex<_Ch_type, _Rx_traits >::assign (
 initializer_list<_Ch_type > __l,
 flag_type __flags = ECMAScript) [inline]
```

Assigns a new regular expression to a regex object.

## Parameters

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| <code>__l</code>     | An initializer list representing a regular expression. |
| <code>__flags</code> | Syntax option flags.                                   |

## Exceptions

|                          |                                                                                                                                                                                              |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> does not contain a valid regular expression pattern interpreted according to <code>__flags</code> . If <code>regex_error</code> is thrown, the object remains unchanged. |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 690 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

## 4.304.3.8 flags()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex<_Ch_type, _Rx_traits >::flags () const [inline]
```

Gets the flags used to construct the regular expression or in the last call to `assign()`.

Definition at line 711 of file `regex.h`.

## 4.304.3.9 getloc()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits >::getloc () const [inline]
```

Gets the locale currently imbued in the regular expression object.

Definition at line 733 of file `regex.h`.

## 4.304.3.10 imbue()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex<_Ch_type, _Rx_traits >::imbue (
 locale_type __loc) [inline]
```

Imbues the regular expression object with the given locale.

## Parameters

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

Definition at line 721 of file `regex.h`.

**4.304.3.11** `mark_count()`

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex< _Ch_type, _Rx_traits >::mark_count () const [inline]
```

Gets the number of marked subexpressions within the regular expression.

Definition at line 699 of file `regex.h`.

**4.304.3.12** `operator=()` [1/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= (
 const basic_regex< _Ch_type, _Rx_traits > & __rhs) [inline]
```

Assigns one regular expression to another.

Definition at line 532 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

**4.304.3.13** `operator=()` [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= (
 basic_regex< _Ch_type, _Rx_traits > && __rhs) [inline], [noexcept]
```

Move-assigns one regular expression to another.

Definition at line 539 of file `regex.h`.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, and `std::move()`.

**4.304.3.14** `operator=()` [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= (
 const _Ch_type * __p) [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.



## Parameters

|                 |                                                                                             |
|-----------------|---------------------------------------------------------------------------------------------|
| <code>_p</code> | A pointer to the start of a null-terminated C-style string containing a regular expression. |
|-----------------|---------------------------------------------------------------------------------------------|

Definition at line 550 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

## 4.304.3.15 operator=() [ 4 / 5 ]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= (
 initializer_list< _Ch_type > __l) [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

## Parameters

|                  |                       |
|------------------|-----------------------|
| <code>__l</code> | The initializer list. |
|------------------|-----------------------|

## Exceptions

|                          |                                                        |
|--------------------------|--------------------------------------------------------|
| <code>regex_error</code> | if <code>__l</code> is not a valid regular expression. |
|--------------------------|--------------------------------------------------------|

Definition at line 562 of file regex.h.

References `std::basic_regex< _Ch_type, _Rx_traits >::assign()`.

## 4.304.3.16 operator=() [ 5 / 5 ]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits , typename _Alloc >
basic_regex& std::basic_regex< _Ch_type, _Rx_traits >::operator= (
 const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s) [inline]
```

Replaces a regular expression with a new one constructed from a string.

**Parameters**

|                                 |                                                        |
|---------------------------------|--------------------------------------------------------|
| <a href="#"><code>_↵</code></a> | A pointer to a string containing a regular expression. |
| <a href="#"><code>_S</code></a> |                                                        |

Definition at line 573 of file `regex.h`.

References `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

**4.304.3.17 swap()**

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex<_Ch_type, _Rx_traits >::swap (
 basic_regex<_Ch_type, _Rx_traits > & __rhs) [inline]
```

Swaps the contents of two regular expression objects.

**Parameters**

|                                    |                                    |
|------------------------------------|------------------------------------|
| <a href="#"><code>__rhs</code></a> | Another regular expression object. |
|------------------------------------|------------------------------------|

Definition at line 743 of file `regex.h`.

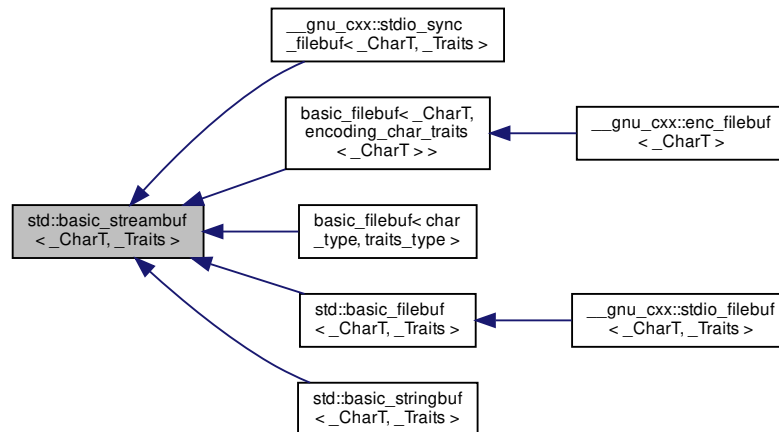
Referenced by `std::basic_regex<_Ch_type, _Rx_traits >::assign()`.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.305 std::basic\_streambuf&lt;\_CharT, \_Traits&gt; Class Template Reference

Inheritance diagram for std::basic\_streambuf<\_CharT, \_Traits>:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Traits` `traits_type`
- typedef `traits_type::int_type` `int_type`
- typedef `traits_type::pos_type` `pos_type`
- typedef `traits_type::off_type` `off_type`
- typedef `basic_streambuf<char_type, traits_type>` `__streambuf_type`

## Public Member Functions

- virtual `~basic_streambuf()`
- `locale getloc()` const
- `streamsize in_avail()`
- `locale pubimbue(const locale &__loc)`
- `int_type sbumpc()`
- `int_type sgetc()`
- `streamsize sgetn(char_type * __s, streamsize __n)`

- [int\\_type](#) [sngetc](#) ()
  - [int\\_type](#) [sputbackc](#) ([char\\_type](#) \_\_c)
  - [int\\_type](#) [sputc](#) ([char\\_type](#) \_\_c)
  - [streamsize](#) [sputn](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
  - [int\\_type](#) [sungetc](#) ()
- 
- [basic\\_streambuf](#) \* [pubsetbuf](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
  - [pos\\_type](#) [pubseekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - [pos\\_type](#) [pubseekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - [int](#) [pubsync](#) ()

#### Protected Member Functions

- [basic\\_streambuf](#) ()
  - **[basic\\_streambuf](#)** (const [basic\\_streambuf](#) &)
  - void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
  - void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
  - void [gbump](#) (int \_\_n)
  - virtual void [imbue](#) (const [locale](#) &\_\_loc)
  - [basic\\_streambuf](#) & **[operator=](#)** (const [basic\\_streambuf](#) &)
  - virtual [int\\_type](#) [overflow](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
  - virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=[traits\\_type::eof](#)())
  - void [pbump](#) (int \_\_n)
  - virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
  - virtual [basic\\_streambuf](#)< [char\\_type](#), [\\_Traits](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
  - void [setg](#) ([char\\_type](#) \* \_\_gbeg, [char\\_type](#) \* \_\_gnext, [char\\_type](#) \* \_\_gend)
  - void [setp](#) ([char\\_type](#) \* \_\_pbeg, [char\\_type](#) \* \_\_pend)
  - virtual [streamsize](#) [showmanyc](#) ()
  - void **[swap](#)** ([basic\\_streambuf](#) &\_\_sb)
  - virtual [int](#) [sync](#) ()
  - virtual [int\\_type](#) [uflow](#) ()
  - virtual [int\\_type](#) [underflow](#) ()
  - virtual [streamsize](#) [xsgetn](#) ([char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
  - virtual [streamsize](#) [xspun](#) (const [char\\_type](#) \*\_\_s, [streamsize](#) \_\_n)
- 
- [char\\_type](#) \* [eback](#) () const
  - [char\\_type](#) \* [gptr](#) () const
  - [char\\_type](#) \* [egptr](#) () const
- 
- [char\\_type](#) \* [pbase](#) () const
  - [char\\_type](#) \* [pptr](#) () const
  - [char\\_type](#) \* [epptr](#) () const

## Protected Attributes

- [locale](#) [\\_M\\_buf\\_locale](#)
- [char\\_type](#) \* [\\_M\\_in\\_beg](#)
- [char\\_type](#) \* [\\_M\\_in\\_cur](#)
- [char\\_type](#) \* [\\_M\\_in\\_end](#)
- [char\\_type](#) \* [\\_M\\_out\\_beg](#)
- [char\\_type](#) \* [\\_M\\_out\\_cur](#)
- [char\\_type](#) \* [\\_M\\_out\\_end](#)

## Friends

- [template](#)<bool [\\_IsMove](#), typename [\\_CharT2](#) >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_char](#)< [\\_CharT2](#) >::\_\_value, [\\_CharT2](#) \* >::\_\_type [\\_\\_copy\\_move\\_a2](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [\\_CharT2](#) \*)
- [streamsize](#) [\\_\\_copy\\_streambufs\\_eof](#) ([basic\\_streambuf](#) \*, [basic\\_streambuf](#) \*, bool &)
- [template](#)<typename [\\_CharT2](#), typename [\\_Distance](#) >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_char](#)< [\\_CharT2](#) >::\_\_value, void >::\_\_type [advance](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#) > &, [\\_Distance](#))
- class [basic\\_ios](#)< [char\\_type](#), [traits\\_type](#) >
- class [basic\\_istream](#)< [char\\_type](#), [traits\\_type](#) >
- class [basic\\_ostream](#)< [char\\_type](#), [traits\\_type](#) >
- [template](#)<typename [\\_CharT2](#) >  
[\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_char](#)< [\\_CharT2](#) >::\_\_value, [istreambuf\\_iterator](#)< [\\_CharT2](#) > >::\_\_type [find](#) ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, const [\\_CharT2](#) &)
- [template](#)<typename [\\_CharT2](#), typename [\\_Traits2](#), typename [\\_Alloc](#) >  
[basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [getline](#) ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [basic\\_string](#)< [\\_CharT2](#), [\\_Traits2](#), [\\_Alloc](#) > &, [\\_CharT2](#))
- class [istreambuf\\_iterator](#)< [char\\_type](#), [traits\\_type](#) >
- [template](#)<typename [\\_CharT2](#), typename [\\_Traits2](#) >  
[basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [operator](#)>> ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [\\_CharT2](#) \*)
- [template](#)<typename [\\_CharT2](#), typename [\\_Traits2](#), typename [\\_Alloc](#) >  
[basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > & [operator](#)>> ([basic\\_istream](#)< [\\_CharT2](#), [\\_Traits2](#) > &, [basic\\_string](#)< [\\_CharT2](#), [\\_Traits2](#), [\\_Alloc](#) > &)
- class [ostreambuf\\_iterator](#)< [char\\_type](#), [traits\\_type](#) >

## 4.305.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_streambuf<_CharT, _Traits>
```

The actual work of input and output (interface).

## Template Parameters

|                         |                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                                      |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits</a> < <a href="#">_CharT</a> >. |

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.

Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
  - The controlled input sequence can be not readable.
  - The controlled output sequence can be not writable.
  - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
  - The controlled sequences can support operations *directly* to or from associated sequences.
  - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.
2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
  - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
  - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
  - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
  - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an output sequence, then a *write position* is available. In this case, *\*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
  - If *xnext* is not a null pointer and  $xbeg < xnext$  for an input sequence, then a *putback position* is available. In this case, *xnext[-1]* shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
  - If *xnext* is not a null pointer and  $xnext < xend$  for an input sequence, then a *read position* is available. In this case, *\*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

Definition at line 80 of file iosfwd.

#### 4.305.2 Member Typedef Documentation

#### 4.305.2.1 \_\_streambuf\_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__streambuf_type
```

This is a non-standard type.

Definition at line 140 of file streambuf.

#### 4.305.2.2 char\_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file streambuf.

#### 4.305.2.3 int\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file streambuf.

#### 4.305.2.4 off\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 135 of file streambuf.

#### 4.305.2.5 pos\_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 134 of file streambuf.

#### 4.305.2.6 traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file streambuf.

### 4.305.3 Constructor & Destructor Documentation

#### 4.305.3.1 ~basic\_streambuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~~basic_streambuf () [inline], [virtual]
```

Destructor deallocates no buffer space.

Definition at line 204 of file streambuf.

#### 4.305.3.2 basic\_streambuf()

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf () [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the basic\_streambuf class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

Definition at line 470 of file streambuf.



## 4.305.4 Member Function Documentation

## 4.305.4.1 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 489 of file streambuf.

## 4.305.4.2 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 4.305.4.3 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 4.305.4.4 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected]
```

Moving the read position.

##### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 4.305.4.5 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

## 4.305.4.6 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

## 4.305.4.7 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf<_CharT, _Traits>::imbue (
 const locale & __loc) [inline], [protected], [virtual]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, and `std::basic_filebuf<char_type, traits_type>`.

Definition at line 583 of file streambuf.

#### 4.305.4.8 `in_avail()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline]
```

Looking ahead into the stream.

##### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

#### 4.305.4.9 `overflow()`

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

##### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

##### Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented in `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_type, traits_type, _Alloc >`, `std::basic_filebuf< char_type, traits_type >`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits, _Alloc >`.

Definition at line 775 of file `streambuf`.

## 4.305.4.10 pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

## Parameters

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), [std::basic\\_filebuf<char\\_type, traits\\_type>](#), [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#), and [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits, \\_Alloc>](#).

Definition at line 731 of file `streambuf`.

## 4.305.4.11 pbase()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

## 4.305.4.12 pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**4.305.4.13 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**4.305.4.14 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 4.305.4.15 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 4.305.4.16 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

## 4.305.4.17 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 4.305.4.18 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 4.305.4.19 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline]
```

Getting the next character.

##### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.305.4.20 `seekoff()`

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 609 of file `streambuf`.



## 4.305.4.21 seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 621 of file `streambuf`.

## 4.305.4.22 setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf<char_type,_Traits>* std::basic_streambuf< _CharT, _Traits >::setbuf (
 char_type * ,
 streamsize) [inline], [protected], [virtual]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

## Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 598 of file `streambuf`.

## 4.305.4.23 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

**4.305.4.24 setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file `streambuf`.

**4.305.4.25 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

## 4.305.4.26 sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sgetn (
 char_type * __s,
 streamsize __n) [inline]
```

Entry point for xsgetn.

## Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns xsgetn(\_\_s,\_\_n). The effect is to fill \_\_s[0] through \_\_s[\_\_n-1] with characters from the input sequence, if possible.

Definition at line 364 of file streambuf.

## 4.305.4.27 showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected],
[virtual]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 656 of file streambuf.

**4.305.4.28   snextc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**4.305.4.29   sputbackc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
 char_type __c) [inline]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.305.4.30   sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline]
```

Entry point for all single-character output functions.

## Parameters

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

## Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

## 4.305.4.31 sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline]
```

Entry point for all single-character output functions.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

## 4.305.4.32 sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**4.305.4.33 sync()**

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync (
 void) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented in `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_type, traits_type >`, `std::basic_filebuf< char_type, traits_type >`, and `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 634 of file `streambuf`.

**4.305.4.34 uflow()**

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

## 4.305.4.35 underflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf<_CharT, _Traits>::underflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

## Note

Base class version does nothing, returns eof().

Reimplemented in `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`, `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_type, traits_type>`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>`, and `__gnu_cxx::stdio_sync_filebuf<_CharT, traits_type>`.

Definition at line 694 of file streambuf.

## 4.305.4.36 xsggetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf<_CharT, _Traits>::xsggetn (
 char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 46 of file `streambuf.tcc`.

**4.305.4.37 xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 80 of file `streambuf.tcc`.

**4.305.5 Member Data Documentation**



#### 4.305.5.1 \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected]
```

Current locale setting.

Definition at line 199 of file streambuf.

#### 4.305.5.2 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 4.305.5.3 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected]
```

Current read area.

Definition at line 192 of file streambuf.

#### 4.305.5.4 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected]
```

End of get area.

Definition at line 193 of file streambuf.

#### 4.305.5.5 \_M\_out\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.305.5.6 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.305.5.7 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected]
```

End of put area.

Definition at line 196 of file streambuf.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf](#)
- [streambuf.tcc](#)

### 4.306 `std::basic_string< _CharT, _Traits, _Alloc >` Class Template Reference

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, basic\_string >` **const\_iterator**
- typedef `_CharT_alloc_traits::const_pointer` **const\_pointer**
- typedef `const value_type &` **const\_reference**
- typedef `std::reverse\_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_CharT_alloc_traits::difference_type` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, basic\_string >` **iterator**
- typedef `_CharT_alloc_traits::pointer` **pointer**
- typedef `value_type &` **reference**
- typedef `std::reverse\_iterator< iterator >` **reverse\_iterator**
- typedef `_CharT_alloc_traits::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- [basic\\_string](#) () noexcept
- [basic\\_string](#) (const \_Alloc &\_\_a)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- [basic\\_string](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n, const \_Alloc &\_\_a)
- [basic\\_string](#) (const \_CharT \*\_\_s, size\_type \_\_n, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (const \_CharT \*\_\_s, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) (size\_type \_\_n, \_CharT \_\_c, const \_Alloc &\_\_a=\_Alloc())
- [basic\\_string](#) ([basic\\_string](#) &&\_\_str) noexcept
- [basic\\_string](#) (initializer\_list<\_CharT> \_\_l, const \_Alloc &\_\_a=\_Alloc())
- **basic\_string** (const [basic\\_string](#) &\_\_str, const \_Alloc &\_\_a)
- **basic\_string** ([basic\\_string](#) &&\_\_str, const \_Alloc &\_\_a)
- template<class \_InputIterator >  
[basic\\_string](#) (\_InputIterator \_\_beg, \_InputIterator \_\_end, const \_Alloc &\_\_a=\_Alloc())
- [~basic\\_string](#) () noexcept
- template<typename \_InputIterator >  
[basic\\_string](#)<\_CharT, \_Traits, \_Alloc> & **M\_replace\_dispatch** (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2, \_\_false\_type)
- template<typename \_InIterator >  
\_CharT \* **S\_construct** (\_InIterator \_\_beg, \_InIterator \_\_end, const \_Alloc &\_\_a, [forward\\_iterator\\_tag](#))
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [append](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [append](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [append](#) (size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & [append](#) (initializer\_list<\_CharT> \_\_l)
- template<class \_InputIterator >  
[basic\\_string](#) & [append](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & [assign](#) ([basic\\_string](#) &&\_\_str) noexcept([allocator\\_traits](#)<\_Alloc>::is\_always\_equal::value)
- [basic\\_string](#) & [assign](#) (const [basic\\_string](#) &\_\_str, size\_type \_\_pos, size\_type \_\_n=[npos](#))
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & [assign](#) (const \_CharT \*\_\_s)
- [basic\\_string](#) & [assign](#) (size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator >  
[basic\\_string](#) & [assign](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- [basic\\_string](#) & [assign](#) (initializer\_list<\_CharT> \_\_l)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [at](#) (size\_type \_\_n)
- reference [back](#) ()
- const\_reference [back](#) () const noexcept
- iterator [begin](#) ()
- const\_iterator [begin](#) () const noexcept
- const \_CharT \* [c\\_str](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept

- int `compare` (const `basic_string` &\_\_str) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- int `compare` (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=npos) const
- int `compare` (const `_CharT` \*\_\_s) const noexcept
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \*\_\_s) const
- int `compare` (size\_type \_\_pos, size\_type \_\_n1, const `_CharT` \*\_\_s, size\_type \_\_n2) const
- size\_type `copy` (`_CharT` \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- `const_reverse_iterator` `crbegin` () const noexcept
- `const_reverse_iterator` `crend` () const noexcept
- const `_CharT` \* `data` () const noexcept
- bool `empty` () const noexcept
- iterator `end` ()
- const\_iterator `end` () const noexcept
- `basic_string` & `erase` (size\_type \_\_pos=0, size\_type \_\_n=npos)
- iterator `erase` (iterator \_\_position)
- iterator `erase` (iterator \_\_first, iterator \_\_last)
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_first_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_not_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_first_of` (const `_CharT` \*\_\_s, size\_type \_\_pos=0) const noexcept
- size\_type `find_first_of` (`_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type `find_last_not_of` (const `basic_string` &\_\_str, size\_type \_\_pos=npos) const noexcept
- size\_type `find_last_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_last_not_of` (const `_CharT` \*\_\_s, size\_type \_\_pos=npos) const noexcept
- size\_type `find_last_not_of` (`_CharT` \_\_c, size\_type \_\_pos=npos) const noexcept
- size\_type `find_last_of` (const `basic_string` &\_\_str, size\_type \_\_pos=npos) const noexcept
- size\_type `find_last_of` (const `_CharT` \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type `find_last_of` (const `_CharT` \*\_\_s, size\_type \_\_pos=npos) const noexcept
- size\_type `find_last_of` (`_CharT` \_\_c, size\_type \_\_pos=npos) const noexcept
- reference `front` ()
- const\_reference `front` () const noexcept
- allocator\_type `get_allocator` () const noexcept
- void `insert` (iterator \_\_p, size\_type \_\_n, `_CharT` \_\_c)
- template<class `_InputIterator` >  
void `insert` (iterator \_\_p, `_InputIterator` \_\_beg, `_InputIterator` \_\_end)
- void `insert` (iterator \_\_p, `initializer_list`< `_CharT` > \_\_l)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n=npos)
- `basic_string` & `insert` (size\_type \_\_pos, const `_CharT` \*\_\_s, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const `_CharT` \*\_\_s)
- `basic_string` & `insert` (size\_type \_\_pos, size\_type \_\_n, `_CharT` \_\_c)
- iterator `insert` (iterator \_\_p, `_CharT` \_\_c)
- size\_type `length` () const noexcept

- size\_type max\_size () const noexcept
- basic\_string & operator+= (const basic\_string &\_\_str)
- basic\_string & operator+= (const \_CharT \*\_\_s)
- basic\_string & operator+= (\_CharT \_\_c)
- basic\_string & operator+= (initializer\_list<\_CharT> \_\_l)
- basic\_string & operator= (const basic\_string &\_\_str)
- basic\_string & operator= (const \_CharT \*\_\_s)
- basic\_string & operator= (\_CharT \_\_c)
- basic\_string & operator= (basic\_string &&\_\_str) noexcept(*/\*conditional \*/*)
- basic\_string & operator= (initializer\_list<\_CharT> \_\_l)
- const\_reference operator[] (size\_type \_\_pos) const noexcept
- reference operator[] (size\_type \_\_pos)
- void pop\_back ()
- void push\_back (\_CharT \_\_c)
- reverse\_iterator rbegin ()
- const\_reverse\_iterator rbegin () const noexcept
- reverse\_iterator rend ()
- const\_reverse\_iterator rend () const noexcept
- basic\_string & replace (size\_type \_\_pos, size\_type \_\_n, const basic\_string &\_\_str)
- basic\_string & replace (size\_type \_\_pos1, size\_type \_\_n1, const basic\_string &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=npos)
- basic\_string & replace (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- basic\_string & replace (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- basic\_string & replace (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, const basic\_string &\_\_str)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<class \_InputIterator>  
basic\_string & replace (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- basic\_string & replace (iterator \_\_i1, iterator \_\_i2, initializer\_list<\_CharT> \_\_l)
- void reserve (size\_type \_\_res\_arg=0)
- void resize (size\_type \_\_n, \_CharT \_\_c)
- void resize (size\_type \_\_n)
- size\_type rfind (const basic\_string &\_\_str, size\_type \_\_pos=npo) const noexcept
- size\_type rfind (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const noexcept
- size\_type rfind (const \_CharT \*\_\_s, size\_type \_\_pos=npo) const noexcept
- size\_type rfind (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- void shrink\_to\_fit () noexcept
- size\_type size () const noexcept
- basic\_string substr (size\_type \_\_pos=0, size\_type \_\_n=npo) const
- void swap (basic\_string &\_\_s) noexcept(*/\*conditional \*/*)

#### Static Public Attributes

- static const size\_type npos

## Protected Types

- typedef iterator `__const_iterator`

## 4.306.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >
```

Managing sequences of characters and character-like objects.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#). Of the [optional sequence requirements](#), only `push_back`, `at`, and array access are supported.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation↵_style.html)

Documentation? What's that? Nathan Myers [ncm@cantrip.org](mailto:ncm@cantrip.org).

A string looks like this:

```

[basic_string<char_type>] [_Rep]
_M_dataplus _M_length
_M_p -----> _M_capacity
 _M_refcount
 unnamed array of char_type
```

Where the `_M_p` points to the first character in the string, and you cast it to a pointer-to-`_Rep` and subtract 1 to get a pointer to the header.

This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: `_Rep::_M_data()`, and `string↵::_M_rep()`; and the allocation function which gets a block of raw bytes and with room enough and constructs a `_Rep` object at the front.

The reason you want `_M_data` pointing to the character array and not the `_Rep` is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the `_Rep` for the debugger to use, so users can check the actual string length.)

Note that the `_Rep` object is a POD so that you can have a static *empty string* `_Rep` object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string `_Rep` object.

All but the last paragraph is considered pretty conventional for a C++ string implementation.

Definition at line 3139 of file `basic_string.h`.

## 4.306.2 Constructor &amp; Destructor Documentation

## 4.306.2.1 basic\_string() [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string () [inline], [noexcept]
```

Default constructor creates an empty string.

Definition at line 3546 of file basic\_string.h.

Referenced by std::basic\_string< char >::substr().

## 4.306.2.2 basic\_string() [2/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const _Alloc & __a) [explicit]
```

Construct an empty string using allocator *a*.

Definition at line 620 of file basic\_string.tcc.

## 4.306.2.3 basic\_string() [3/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Construct string with copy of value of *str*.

## Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

Definition at line 612 of file basic\_string.tcc.

## 4.306.2.4 basic\_string() [4/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
```

```
const basic_string< _CharT, _Traits, _Alloc > & __str,
size_type __pos,
const _Alloc & __a = _Alloc())
```

Construct string as copy of a substring.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__a</code>   | Allocator to use.                      |

Definition at line 626 of file `basic_string.tcc`.

#### 4.306.2.5 `basic_string()` [5/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n)
```

Construct string as copy of a substring.

#### Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |

Definition at line 636 of file `basic_string.tcc`.

#### 4.306.2.6 `basic_string()` [6/12]

```
template<typename _CharT , typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n,
 const _Alloc & __a)
```

Construct string as copy of a substring.



## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__str</code> | Source string.                         |
| <code>__pos</code> | Index of first character to copy from. |
| <code>__n</code>   | Number of characters to copy.          |
| <code>__a</code>   | Allocator to use.                      |

Definition at line 646 of file `basic_string.tcc`.

## 4.306.2.7 basic\_string() [7/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 size_type __n,
 const _Alloc & __a = _Alloc())
```

Construct string initialized by a character array.

## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source character array.                          |
| <code>__n</code> | Number of characters to copy.                    |
| <code>__a</code> | Allocator to use (default is default allocator). |

NB: `__s` must have at least `__n` characters, `'\0'` has no special meaning.

Definition at line 658 of file `basic_string.tcc`.

## 4.306.2.8 basic\_string() [8/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc >::basic_string (
 const _CharT * __s,
 const _Alloc & __a = _Alloc()) [inline]
```

Construct string as copy of a C string.

## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__s</code> | Source C string.                                 |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 3619 of file basic\_string.h.

#### 4.306.2.9 basic\_string() [9/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 size_type __n,
 _CharT __c,
 const _Alloc & __a = _Alloc())
```

Construct string as multiple characters.

##### Parameters

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>↔<br/>_n</code> | Number of characters.                            |
| <code>↔<br/>_c</code> | Character to use.                                |
| <code>↔<br/>_a</code> | Allocator to use (default is default allocator). |

Definition at line 664 of file basic\_string.tcc.

#### 4.306.2.10 basic\_string() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move construct string.

##### Parameters

|                    |                |
|--------------------|----------------|
| <code>__str</code> | Source string. |
|--------------------|----------------|

The newly-created string contains the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 3640 of file basic\_string.h.

#### 4.306.2.11 basic\_string() [11/12]

```
template<typename _CharT, typename _Traits , typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
```

```
initializer_list<_CharT> __l,
const _Alloc & __a = _Alloc())
```

Construct string from an initializer list.

#### Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | std::initializer_list of characters.             |
| <code>__a</code> | Allocator to use (default is default allocator). |

Definition at line 679 of file basic\_string.tcc.

#### 4.306.2.12 basic\_string() [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _InputIterator>
std::basic_string<_CharT, _Traits, _Alloc>::basic_string (
 _InputIterator __beg,
 _InputIterator __end,
 const _Alloc & __a = _Alloc())
```

Construct string as copy of a range.

#### Parameters

|                    |                                                  |
|--------------------|--------------------------------------------------|
| <code>__beg</code> | Start of range.                                  |
| <code>__end</code> | End of range.                                    |
| <code>__a</code>   | Allocator to use (default is default allocator). |

Definition at line 672 of file basic\_string.tcc.

#### 4.306.2.13 ~basic\_string()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string<_CharT, _Traits, _Alloc>::~~basic_string () [inline], [noexcept]
```

Destroy the string instance.

Definition at line 3717 of file basic\_string.h.

### 4.306.3 Member Function Documentation

**4.306.3.1** `append()` [1/7]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 769 of file `basic_string.tcc`.

Referenced by `std::basic_string< char >::append()`, and `std::basic_string< char >::operator+=()`.

**4.306.3.2** `append()` [2/7]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos)
```

Append a substring.

**Parameters**

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of

available characters in `__str`, the remainder of `__str` is appended.

Definition at line 786 of file `basic_string.tcc`.

#### 4.306.3.3 `append()` [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s,
 size_type __n)
```

Append a C substring.

##### Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__s</code> | The C string to append.             |
| <code>__n</code> | The number of characters to append. |

##### Returns

Reference to this string.

Definition at line 742 of file `basic_string.tcc`.

#### 4.306.3.4 `append()` [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::append (
 const _CharT * __s) [inline]
```

Append a C string.

##### Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

##### Returns

Reference to this string.



Definition at line 4252 of file `basic_string.h`.

**4.306.3.5 append()** [5/7]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
 size_type __n,
 _CharT __c)
```

Append multiple characters.

**Parameters**

|                                                                                                    |                                     |
|----------------------------------------------------------------------------------------------------|-------------------------------------|
|  <code>__n</code> | The number of characters to append. |
|  <code>__c</code> | The character to use.               |

**Returns**

Reference to this string.

Appends `__n` copies of `__c` to this string.


Definition at line 725 of file `basic_string.tcc`.

**4.306.3.6 append()** [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (
 initializer_list< _CharT > __l) [inline]
```

Append an `initializer_list` of characters.

**Parameters**

|                                                                                                      |                                                            |
|------------------------------------------------------------------------------------------------------|------------------------------------------------------------|
|  <code>__l</code> | The <code>initializer_list</code> of characters to append. |
|------------------------------------------------------------------------------------------------------|------------------------------------------------------------|

**Returns**

Reference to this string.

Definition at line 4276 of file `basic_string.h`.

#### 4.306.3.7 append() [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::append (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Append a range of characters.

##### Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

##### Returns

Reference to this string.

Appends characters in the range [`__first`,`__last`) to this string.

Definition at line 4290 of file `basic_string.h`.

#### 4.306.3.8 assign() [1/8]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Set value to contents of another string.

##### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

##### Returns

Reference to this string.

Definition at line 687 of file `basic_string.tcc`.

Referenced by `std::basic_string< char >::assign()`, and `std::basic_string< char >::operator=()`.

**4.306.3.9** `assign()` [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 4358 of file `basic_string.h`.

**4.306.3.10** `assign()` [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos,
 size_type __n = npos) [inline]
```

Set value to a substring of a string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the



number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 4380 of file `basic_string.h`.

#### 4.306.3.11 `assign()` [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc > & std::basic_string<_CharT, _Traits, _Alloc >::assign (
 const _CharT * __s,
 size_type __n)
```

Set value to a C substring.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | The C string to use.         |
| <code>__n</code> | Number of characters to use. |

##### Returns

Reference to this string.

This function sets the value of this string to the first `__n` characters of `__s`. If `__n` is larger than the number of available characters in `__s`, the remainder of `__s` is used.

Definition at line 703 of file `basic_string.tcc`.

#### 4.306.3.12 `assign()` [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign (
 const _CharT * __s) [inline]
```

Set value to contents of a C string.

##### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | The C string to use. |
|------------------|----------------------|

**Returns**

Reference to this string.

This function sets the value of this string to the value of `__s`. The data is copied, so there is no dependence on `__s` once the function returns.

Definition at line 4408 of file `basic_string.h`.

**4.306.3.13 assign()** [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
 size_type __n,
 _CharT __c) [inline]
```

Set value to multiple characters.

**Parameters**

|                  |                                 |
|------------------|---------------------------------|
| <code>__n</code> | Length of the resulting string. |
| <code>__c</code> | The character to use.           |

**Returns**

Reference to this string.

This function sets the value of this string to `__n` copies of character `__c`.

Definition at line 4424 of file `basic_string.h`.

**4.306.3.14 assign()** [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Set value to a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to append. |
| <code>__last</code>  | Iterator marking the end of the range.              |

**Returns**

Reference to this string.

Sets value of string to characters in the range [`__first`,`__last`).

Definition at line 4437 of file `basic_string.h`.

**4.306.3.15 assign()** [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::assign (
 initializer_list<_CharT > __l) [inline]
```

Set value to an `initializer_list` of characters.

**Parameters**

|                |                                                            |
|----------------|------------------------------------------------------------|
| <code>↔</code> | The <code>initializer_list</code> of characters to assign. |
| <code>↔</code> |                                                            |
| <code>↔</code> |                                                            |
| <code>↔</code> |                                                            |
| <code>/</code> |                                                            |

**Returns**

Reference to this string.

Definition at line 4447 of file `basic_string.h`.

**4.306.3.16 at()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc >::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>↔</code>   | The index of the character to access. |
| <code>__n</code> |                                       |

**Returns**

Read-only (const) reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 4080 of file `basic_string.h`.

**4.306.3.17 at()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
 size_type __n) [inline]
```

Provides access to the data contained in the string.

**Parameters**

|                 |                                       |
|-----------------|---------------------------------------|
| <code>_↔</code> | The index of the character to access. |
| <code>_n</code> |                                       |

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 4102 of file `basic_string.h`.

**4.306.3.18 back()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back () [inline]
```

Returns a read/write reference to the data at the last element of the string.

Definition at line 4141 of file basic\_string.h.

#### 4.306.3.19 back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 4152 of file basic\_string.h.

#### 4.306.3.20 begin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () [inline]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Definition at line 3803 of file basic\_string.h.

Referenced by std::basic\_string< char >::crend(), std::regex\_match(), std::regex\_replace(), std::regex\_search(), and std::basic\_string< char >::rend().

#### 4.306.3.21 begin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 3814 of file basic\_string.h.

#### 4.306.3.22 c\_str()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Definition at line 5207 of file basic\_string.h.

Referenced by std::collate< \_CharT >::do\_compare(), std::money\_get< \_CharT, \_Inlter >::do\_get(), std::num\_get< \_CharT, \_Inlter >::do\_get(), std::collate< \_CharT >::do\_transform(), std::regex\_replace(), and std::experimental::filesystem::v1::filesystem\_error::what().

#### 4.306.3.23 capacity()

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 3976 of file basic\_string.h.

Referenced by `std::basic_string< char >::push_back()`, and `std::basic_string< char >::shrink_to_fit()`.

#### 4.306.3.24 cbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

Definition at line 3878 of file basic\_string.h.

#### 4.306.3.25 cend()

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 3886 of file basic\_string.h.

#### 4.306.3.26 clear()

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear () [inline], [noexcept]
```

Erases the string, making it empty.

Definition at line 4004 of file basic\_string.h.

#### 4.306.3.27 compare() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 const basic_string< _CharT, _Traits, _Alloc > & __str) const [inline]
```

Compare to a string.

## Parameters

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

## Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 5763 of file `basic_string.h`.

## 4.306.3.28 compare() [2/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 size_type __pos,
 size_type __n,
 const basic_string< _CharT, _Traits, _Alloc > & __str) const
```

Compare substring to a string.

## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

## Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer < 0 if the substring is ordered before `__str`, 0 if their values are equivalent, or > 0 if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1388 of file `basic_string.tcc`.

**4.306.3.29 compare()** [3/6]

```
template<typename _CharT , typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos2,
 size_type __n2 = npos) const
```

Compare substring to a substring.

**Parameters**

|                     |                                               |
|---------------------|-----------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.        |
| <code>__n1</code>   | Number of characters in substring.            |
| <code>__str</code>  | String to compare against.                    |
| <code>__pos2</code> | Index of first character of substring of str. |
| <code>__n2</code>   | Number of characters in substring of str.     |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer < 0 if this substring is ordered before the substring of `__str`, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1403 of file `basic_string.tcc`.

**4.306.3.30 compare()** [4/6]

```
template<typename _CharT, typename _Traits , typename _Alloc >
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
 const _CharT * __s) const [noexcept]
```

Compare to a C string.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__s</code> | C string to compare against. |
|------------------|------------------------------|



**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before \_\_s, 0 if their values are equivalent, or > 0 if this string is ordered after \_\_s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from \_\_s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1421 of file basic\_string.tcc.

**4.306.3.31 compare()** [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) const
```

Compare substring to a C string.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | C string to compare against.           |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the \_\_n1 characters starting at pos. Returns an integer < 0 if the substring is ordered before \_\_s, 0 if their values are equivalent, or > 0 if the substring is ordered after \_\_s. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from \_\_s. The function then compares the two string by calling traits::compare(substring.data(),\_\_s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1436 of file basic\_string.tcc.

**4.306.3.32 compare()** [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string<_CharT, _Traits, _Alloc>::compare (
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2) const
```

Compare substring against a character array.

**Parameters**

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n1</code>  | Number of characters in substring.     |
| <code>__s</code>   | character array to compare against.    |
| <code>__n2</code>  | Number of characters of s.             |

**Returns**

Integer < 0, 0, or > 0.

Form the substring of this string from the `__n1` characters starting at `__pos`. Form a string from the first `__n2` characters of `__s`. Returns an integer < 0 if this substring is ordered before the string from `__s`, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from `__s`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__n2`. The function then compares the two strings by calling `traits::compare(substring.data(),s,rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

NB: `s` must have at least `n2` characters, `'\0'` has no special meaning.

Definition at line 1452 of file `basic_string.tcc`.

**4.306.3.33 copy()**

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
 _CharT * __s,
 size_type __n,
 size_type __pos = 0) const
```

Copy substring into C string.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__s</code>   | C string to copy value into.      |
| <code>__n</code>   | Number of characters to copy.     |
| <code>__pos</code> | Index of first character to copy. |

**Returns**

Number of characters actually copied

**Exceptions**

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Copies up to `__n` characters starting at `__pos` into the C string `__s`. If `__pos` is greater than `size()`, `out_of_range` is thrown.

Definition at line 1138 of file `basic_string.tcc`.

#### 4.306.3.34 `crbegin()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 3895 of file `basic_string.h`.

#### 4.306.3.35 `crend()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 3904 of file `basic_string.h`.

#### 4.306.3.36 `data()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT* std::basic_string<_CharT, _Traits, _Alloc>::data () const [inline], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

Definition at line 5219 of file `basic_string.h`.

Referenced by `std::basic_regex<_Ch_type, _Rx_traits>::assign()`, `std::basic_string<char>::compare()`, `std::collate<_CharT>::do_compare()`, `std::collate<_CharT>::do_transform()`, `std::match_results<_Bi_iter>::format()`, `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::from_bytes()`, `std::wstring_convert<_Codecvt, _Elem, _Wide_alloc, _Byte_alloc>::to_bytes()`, and `std::regex_traits<_CharType>::transform()`.

**4.306.3.37 empty()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string<_CharT, _Traits, _Alloc>::empty () const [inline], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 4026 of file `basic_string.h`.

**4.306.3.38 end()** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string<_CharT, _Traits, _Alloc>::end () [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Definition at line 3822 of file `basic_string.h`.

Referenced by `std::basic_string< char >::rbegin()`, `std::basic_string< char >::rbegin()`, `std::regex_match()`, `std::regex_replace()`, and `std::regex_search()`.

**4.306.3.39 end()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string<_CharT, _Traits, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

Definition at line 3833 of file `basic_string.h`.

**4.306.3.40 erase()** [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::erase (
 size_type __pos = 0,
 size_type __n = npos) [inline]
```

Remove characters.

**Parameters**

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| <code>__pos</code> | Index of first character to remove (default 0).     |
| <code>__n</code>   | Number of characters to remove (default remainder). |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is beyond the end of this string. |
|--------------------------------|-------------------------------------------------|

Removes *\_\_n* characters from this string starting at *\_\_pos*. The length of the string is reduced by *\_\_n*. If there are *< \_\_n* characters to remove, the remainder of the string is truncated. If *\_\_p* is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4710 of file `basic_string.h`.

**4.306.3.41 erase()** [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
 iterator __position) [inline]
```

Remove one character.

**Parameters**

|                   |                                               |
|-------------------|-----------------------------------------------|
| <i>__position</i> | Iterator referencing the character to remove. |
|-------------------|-----------------------------------------------|

**Returns**

iterator referencing same location after removal.

Removes the character at *\_\_position* from this string. The value of the string doesn't change if an error is thrown.

Definition at line 4726 of file `basic_string.h`.

**4.306.3.42 erase()** [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >←
::erase (
 iterator __first,
 iterator __last)
```

Remove a range of characters.

**Parameters**

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

**Returns**

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

Definition at line 835 of file basic\_string.tcc.

**4.306.3.43 find()** [1/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a C substring.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

**Returns**

Index of start of first occurrence.

Starting from `__pos`, searches forward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1191 of file basic\_string.tcc.

Referenced by `std::basic_string< char >::find()`, and `std::basic_string< char >::find_first_of()`.

## 4.306.3.44 find() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find (
 const basic_string<_CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5271 of file `basic_string.h`.

## 4.306.3.45 find() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string to locate.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5286 of file `basic_string.h`.

**4.306.3.46** `find()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1227 of file `basic_string.tcc`.

**4.306.3.47** `find_first_not_of()` [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5579 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_first_not_of()`.



## 4.306.3.48 find\_first\_not\_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character not in C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.                |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>__s</code> to consider. |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1322 of file `basic_string.tcc`.

## 4.306.3.49 find\_first\_not\_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_not_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character not in C string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.       |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5610 of file basic\_string.h.

#### 4.306.3.50 find\_first\_not\_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
 _CharT __c,
 size_type __pos = 0) const [noexcept]
```

Find position of a different character.

##### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to avoid.                            |
| <code>__pos</code> | Index of character to search from (default 0). |

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1335 of file basic\_string.tcc.

#### 4.306.3.51 find\_first\_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of string.

##### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5412 of file `basic_string.h`.

Referenced by `std::basic_string<char>::find_first_of()`.

#### 4.306.3.52 find\_first\_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::find_first_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find position of a character of C substring.

##### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.                 |
| <code>__pos</code> | Index of character to search from.                      |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

##### Returns

Index of first occurrence.

Starting from `__pos`, searches forward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1284 of file `basic_string.tcc`.

#### 4.306.3.53 find\_first\_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc >::find_first_of (
 const _CharT * __s,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character of C string.

##### Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__s</code>   | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5443 of file `basic_string.h`.

**4.306.3.54 find\_first\_of()** [ 4 / 4 ]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
 _CharT __c,
 size_type __pos = 0) const [inline], [noexcept]
```

Find position of a character.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__c</code>   | Character to locate.                           |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for the character `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `find(__c, __pos)`.

Definition at line 5463 of file `basic_string.h`.

**4.306.3.55 find\_last\_not\_of()** [ 1 / 4 ]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5661 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_not_of()`.

**4.306.3.56 find\_last\_not\_of()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character not in C substring.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from.               |
| <code>__n</code>   | Number of characters from <code>s</code> to consider. |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1346 of file `basic_string.tcc`.

**4.306.3.57 find\_last\_not\_of()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character not in C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to avoid.              |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5692 of file `basic_string.h`.

**4.306.3.58 find\_last\_not\_of()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a different character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to avoid.                                   |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for a character other than `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1368 of file `basic_string.tcc`.

**4.306.3.59 find\_last\_of()** [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5496 of file `basic_string.h`.

Referenced by `std::basic_string< char >::find_last_of()`.

## 4.306.3.60 find\_last\_of() [2/4]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a character of C substring.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the first `__n` characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1300 of file `basic_string.tcc`.

**4.306.3.61** `find_last_of()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character of C string.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__s</code>   | C string containing characters to locate.             |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__s` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5527 of file `basic_string.h`.

**4.306.3.62** `find_last_of()` [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::find_last_of (
 _CharT __c,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Note: equivalent to `rfind(__c, __pos)`.

Definition at line 5547 of file `basic_string.h`.



**4.306.3.63** front() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string<_CharT, _Traits, _Alloc>::front () [inline]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 4119 of file basic\_string.h.

**4.306.3.64** front() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc>::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 4130 of file basic\_string.h.

**4.306.3.65** get\_allocator()

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string<_CharT, _Traits, _Alloc>::get_allocator () const [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

Definition at line 5241 of file basic\_string.h.

Referenced by std::basic\_string<char>::assign(), std::basic\_string<char>::basic\_string(), std::basic\_string<char>::clear(), std::wstring\_convert<\_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc>::from\_bytes(), std::operator+(), std::wstring\_convert<\_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc>::to\_bytes(), and std::basic\_string<char>::~basic\_string().

**4.306.3.66** insert() [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::insert (
 iterator __p,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4497 of file `basic_string.h`.

Referenced by `std::basic_string<char>::insert()`.

**4.306.3.67 insert()** [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline]
```

Insert a range of characters.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg,__end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4514 of file `basic_string.h`.

## 4.306.3.68 insert() [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::insert (
 iterator __p,
 initializer_list<_CharT > __l) [inline]
```

Insert an initializer\_list of characters.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__l</code> | The initializer_list of characters to insert.         |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Definition at line 4525 of file basic\_string.h.

## 4.306.3.69 insert() [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Alloc > & __str) [inline]
```

Insert value of a string.

## Parameters

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is

thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4545 of file basic\_string.h.

#### 4.306.3.70 insert() [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos1,
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos2,
 size_type __n = npos) [inline]
```

Insert a substring.

##### Parameters

|                     |                                       |
|---------------------|---------------------------------------|
| <code>__pos1</code> | Position in string to insert at.      |
| <code>__str</code>  | The string to insert.                 |
| <code>__pos2</code> | Start of characters in str to insert. |
| <code>__n</code>    | Number of characters to insert.       |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                                             |
|--------------------------------|-----------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                             |
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4567 of file basic\_string.h.

#### 4.306.3.71 insert() [6/9]

```
template<typename _CharT, typename _Traits , typename _Alloc >
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 const _CharT * __s,
 size_type __n)
```

Insert a C substring.

## Parameters

|                    |                                     |
|--------------------|-------------------------------------|
| <code>__pos</code> | Position in string to insert at.    |
| <code>__s</code>   | The C string to insert.             |
| <code>__n</code>   | The number of characters to insert. |

## Returns

Reference to this string.

## Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts the first `__n` characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 804 of file `basic_string.tcc`.

## 4.306.3.72 insert() [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 const _CharT * __s) [inline]
```

Insert a C string.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__pos</code> | Position in string to insert at. |
| <code>__s</code>   | The C string to insert.          |

## Returns

Reference to this string.

## Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |

Inserts the first  $n$  characters of `__s` starting at `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos` is beyond `end()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4608 of file `basic_string.h`.

#### 4.306.3.73 `insert()` [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::insert (
 size_type __pos,
 size_type __n,
 _CharT __c) [inline]
```

Insert multiple characters.

##### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | Index in string to insert at.  |
| <code>__n</code>   | Number of characters to insert |
| <code>__c</code>   | The character to insert.       |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .         |
| <code>std::out_of_range</code> | If <code>__pos</code> is beyond the end of this string. |

Inserts `__n` copies of character `__c` starting at index `__pos`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos > length()`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4631 of file `basic_string.h`.

#### 4.306.3.74 `insert()` [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
 iterator __p,
 _CharT __c) [inline]
```

Insert one character.

## Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

## Returns

Iterator referencing newly inserted char.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4649 of file `basic_string.h`.

## 4.306.3.75 length()

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::length () const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3919 of file `basic_string.h`.

Referenced by `std::collate<_CharT>::do_compare()`, and `std::collate<_CharT>::do_transform()`.

## 4.306.3.76 max\_size()

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string<_CharT, _Traits, _Alloc>::max_size () const [inline], [noexcept]
```

Returns the `size()` of the largest possible string.

Definition at line 3924 of file `basic_string.h`.

Referenced by `std::getline()`.

## 4.306.3.77 operator+=( ) [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::operator+= (
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 4166 of file `basic_string.h`.

**4.306.3.78 operator+=()** [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator+= (
 const _CharT * __s) [inline]
```

Append a C string.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The C string to append. |
|------------------|-------------------------|

**Returns**

Reference to this string.

Definition at line 4175 of file `basic_string.h`.

**4.306.3.79 operator+=()** [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator+= (
 _CharT __c) [inline]
```

Append a character.

**Parameters**

|                  |                          |
|------------------|--------------------------|
| <code>__c</code> | The character to append. |
|------------------|--------------------------|



**Returns**

Reference to this string.

Definition at line 4184 of file basic\_string.h.

**4.306.3.80 operator+=()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator+= (
 initializer_list<_CharT > __l) [inline]
```

Append an initializer\_list of characters.

**Parameters**

|   |                                                    |
|---|----------------------------------------------------|
| ↩ | The initializer_list of characters to be appended. |
| ↩ |                                                    |
| ↩ |                                                    |
| ↩ |                                                    |
| / |                                                    |

**Returns**

Reference to this string.

Definition at line 4197 of file basic\_string.h.

**4.306.3.81 operator=()** [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator= (
 const basic_string<_CharT, _Traits, _Alloc > & __str) [inline]
```

Assign the value of *str* to this string.

**Parameters**

|       |                |
|-------|----------------|
| __str | Source string. |
|-------|----------------|


Definition at line 3725 of file basic\_string.h.

**4.306.3.82 operator=()** [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 const _CharT * __s) [inline]
```

Copy contents of *s* into this string.

**Parameters**

|                                                                                          |                                |
|------------------------------------------------------------------------------------------|--------------------------------|
| <br>__s | Source null-terminated string. |
|------------------------------------------------------------------------------------------|--------------------------------|


Definition at line 3733 of file basic\_string.h.

**4.306.3.83 operator=()** [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 _CharT __c) [inline]
```

Set value to string of length 1.

**Parameters**

|                                                                                            |                   |
|--------------------------------------------------------------------------------------------|-------------------|
| <br>__c | Source character. |
|--------------------------------------------------------------------------------------------|-------------------|

Assigning to a character makes this string length 1 and `(*this)[0] == c`.

Definition at line 3744 of file basic\_string.h.

**4.306.3.84 operator=()** [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::operator= (
 basic_string< _CharT, _Traits, _Alloc > && __str) [inline], [noexcept]
```

Move assign the value of *str* to this string.

**Parameters**

|       |                |
|-------|----------------|
| __str | Source string. |
|-------|----------------|

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

Definition at line 3759 of file basic\_string.h.

#### 4.306.3.85 operator=() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::operator= (
 initializer_list<_CharT > __l) [inline]
```

Set value to string constructed from initializer list.

##### Parameters

|   |                        |
|---|------------------------|
| ↩ | std::initializer_list. |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |
| ↩ |                        |

Definition at line 3772 of file basic\_string.h.

#### 4.306.3.86 operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string<_CharT, _Traits, _Alloc >::operator[] (
 size_type __pos) const [inline], [noexcept]
```

Subscript access to the data contained in the string.

##### Parameters

|       |                                       |
|-------|---------------------------------------|
| __pos | The index of the character to access. |
|-------|---------------------------------------|

##### Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out\_of\_range lookups are not defined. (For checked lookups see at().)

Definition at line 4041 of file basic\_string.h.

**4.306.3.87 operator[]()** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
 size_type __pos) [inline]
```

Subscript access to the data contained in the string.

**Parameters**

|                    |                                       |
|--------------------|---------------------------------------|
| <code>__pos</code> | The index of the character to access. |
|--------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.) Unshares the string.

Definition at line 4058 of file `basic_string.h`.

**4.306.3.88 pop\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::pop_back () [inline]
```

Remove the last character.

The string must be non-empty.

Definition at line 4755 of file `basic_string.h`.

**4.306.3.89 push\_back()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
 _CharT __c) [inline]
```

Append a single character.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | Character to append. |
|------------------|----------------------|

Definition at line 4331 of file basic\_string.h.

Referenced by std::basic\_string<char>::operator+=( ).

#### 4.306.3.90 rbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 3842 of file basic\_string.h.

#### 4.306.3.91 rbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

Definition at line 3851 of file basic\_string.h.

#### 4.306.3.92 rend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

Definition at line 3860 of file basic\_string.h.

#### 4.306.3.93 rend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string<_CharT, _Traits, _Alloc>::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

Definition at line 3869 of file basic\_string.h.

#### 4.306.3.94 replace() [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
 size_type __pos,
 size_type __n,
 const basic_string<_CharT, _Traits, _Alloc> & __str) [inline]
```

Replace characters with value from another string.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4780 of file `basic_string.h`.

Referenced by `std::basic_string< char >::append()`, `std::basic_string< char >::assign()`, `std::basic_string< char >::insert()`, and `std::basic_string< char >::replace()`.

**4.306.3.95 replace()** [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
 size_type __pos1,
 size_type __n1,
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos2,
 size_type __n2 = npos) [inline]
```

Replace characters with value from another string.

**Parameters**

|                     |                                                |
|---------------------|------------------------------------------------|
| <code>__pos1</code> | Index of first character to replace.           |
| <code>__n1</code>   | Number of characters to be replaced.           |
| <code>__str</code>  | String to insert.                              |
| <code>__pos2</code> | Index of first character of <i>str</i> to use. |
| <code>__n2</code>   | Number of characters from <i>str</i> to use.   |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                                           |
|--------------------------|---------------------------------------------------------------------------|
| <i>std::out_of_range</i> | If <i>__pos1</i> > <i>size()</i> or <i>__pos2</i> > <i>__str.size()</i> . |
| <i>std::length_error</i> | If new length exceeds <i>max_size()</i> .                                 |

Removes the characters in the range [*\_\_pos1*, *\_\_pos1* + *n*) from this string. In place, the value of *\_\_str* is inserted. If *\_\_pos* is beyond end of string, *out\_of\_range* is thrown. If the length of the result exceeds *max\_size()*, *length\_error* is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4802 of file *basic\_string.h*.

**4.306.3.96 replace()** [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string<_CharT, _Traits, _Alloc> & std::basic_string<_CharT, _Traits, _Alloc>::replace
(
 size_type __pos,
 size_type __n1,
 const _CharT * __s,
 size_type __n2)
```

Replace characters with value of a C substring.

**Parameters**

|              |                                            |
|--------------|--------------------------------------------|
| <i>__pos</i> | Index of first character to replace.       |
| <i>__n1</i>  | Number of characters to be replaced.       |
| <i>__s</i>   | C string to insert.                        |
| <i>__n2</i>  | Number of characters from <i>s</i> to use. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                           |
|--------------------------|-------------------------------------------|
| <i>std::out_of_range</i> | If <i>pos1</i> > <i>size()</i> .          |
| <i>std::length_error</i> | If new length exceeds <i>max_size()</i> . |

Removes the characters in the range [*\_\_pos*, *\_\_pos* + *\_\_n1*) from this string. In place, the first *\_\_n2* characters of *\_\_s* are inserted, or all of *\_\_s* if *\_\_n2* is too large. If *\_\_pos* is beyond end of string, *out\_of\_range* is thrown. If the length of

result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 858 of file `basic_string.tcc`.

#### 4.306.3.97 `replace()` [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n1,
 const _CharT * __s) [inline]
```

Replace characters with value of a C string.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__s</code>   | C string to insert.                  |

##### Returns

Reference to this string.

##### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos &gt; size()</code> .               |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[__pos, __pos + __n1)` from this string. In place, the characters of `__s` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4847 of file `basic_string.h`.

#### 4.306.3.98 `replace()` [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
 size_type __pos,
 size_type __n1,
 size_type __n2,
 _CharT __c) [inline]
```

Replace characters with multiple characters.



## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n1</code>  | Number of characters to be replaced. |
| <code>__n2</code>  | Number of characters to insert.      |
| <code>__c</code>   | Character to insert.                 |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> .             |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |

Removes the characters in the range `[pos, pos + n1)` from this string. In place, `__n2` copies of `__c` are inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4871 of file `basic_string.h`.

## 4.306.3.99 replace() [6/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const basic_string<_CharT, _Traits, _Alloc > & __str) [inline]
```

Replace range of characters with string.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

## Returns

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4889 of file `basic_string.h`.

**4.306.3.100 replace()** [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s,
 size_type __n) [inline]
```

Replace range of characters with C substring.

**Parameters**

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.     |
| <code>__i2</code> | Iterator referencing end of range to replace.       |
| <code>__s</code>  | C string value to insert.                           |
| <code>__n</code>  | Number of characters from <code>s</code> to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.




Definition at line 4908 of file `basic_string.h`.

## 4.306.3.101 replace() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s) [inline]
```

Replace range of characters with C string.

## Parameters

|                                                                                                     |                                                 |
|-----------------------------------------------------------------------------------------------------|-------------------------------------------------|
|  <code>__i1</code> | Iterator referencing start of range to replace. |
|  <code>__i2</code> | Iterator referencing end of range to replace.   |
|  <code>__s</code>  | C string value to insert.                       |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.


Definition at line 4929 of file `basic_string.h`.

## 4.306.3.102 replace() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 size_type __n,
 _CharT __c) [inline]
```

Replace range of characters with multiple characters.

## Parameters

|                                                                                                       |                                                 |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|
|  <code>__i1</code> | Iterator referencing start of range to replace. |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|

**Parameters**

|                   |                                               |
|-------------------|-----------------------------------------------|
| <code>__i2</code> | Iterator referencing end of range to replace. |
| <code>__n</code>  | Number of characters to insert.               |
| <code>__c</code>  | Character to insert.                          |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4950 of file `basic_string.h`.

**4.306.3.103 replace()** [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator >
basic_string& std::basic_string< _CharT, _Traits, _Alloc >::replace (
 iterator __i1,
 iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline]
```

Replace range of characters with range.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4974 of file `basic_string.h`.

**4.306.3.104 replace()** [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string& std::basic_string<_CharT, _Traits, _Alloc>::replace (
 iterator __i1,
 iterator __i2,
 initializer_list<_CharT> __l) [inline]
```

Replace range of characters with `initializer_list`.

**Parameters**

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, characters in the range `[__k1, __k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 5043 of file `basic_string.h`.

**4.306.3.105 reserve()**

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
 size_type __res_arg = 0)
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                |                                                             |
|--------------------------------|-------------------------------------------------------------|
| <code>std::length_error</code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 946 of file `basic_string.tcc`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::tr2::operator>>()`, `std::basic_stringbuf< _CharT, _↵ Traits, _Alloc >::overflow()`, `std::basic_string< char >::push_back()`, and `std::basic_string< char >::shrink_to_fit()`.

**4.306.3.106 resize()** [1/2]

```
template<typename _CharT, typename _Traits , typename _Alloc >
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
 size_type __n,
 _CharT __c)
```

Resizes the string to the specified number of characters.

**Parameters**

|                        |                                                 |
|------------------------|-------------------------------------------------|
| <code>↵<br/>__n</code> | Number of characters the string should contain. |
| <code>↵<br/>__c</code> | Character to fill any new elements.             |

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to `__c`.

Definition at line 1085 of file basic\_string.tcc.


Referenced by std::money\_get< \_CharT, \_InIter >::do\_get(), and std::basic\_string< char >::resize().

#### 4.306.3.107 resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
 size_type __n) [inline]
```

Resizes the string to the specified number of characters.

##### Parameters

|                                                                                                       |                                                 |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| <br><code>__n</code> | Number of characters the string should contain. |
|-------------------------------------------------------------------------------------------------------|-------------------------------------------------|

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

Definition at line 3951 of file basic\_string.h.

#### 4.306.3.108 rfind() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const basic_string< _CharT, _Traits, _Alloc > & __str,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a string.

##### Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5333 of file basic\_string.h.

Referenced by `std::basic_string< char >::find_last_of()`, and `std::basic_string< char >::rfind()`.

#### 4.306.3.109 `rfind()` [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
 const _CharT * __s,
 size_type __pos,
 size_type __n) const [noexcept]
```

Find last position of a C substring.

##### Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                     |
| <code>__pos</code> | Index of character to search back from.                 |
| <code>__n</code>   | Number of characters from <code>s</code> to search for. |

##### Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for the first `__n` characters in `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 1245 of file `basic_string.tcc`.

#### 4.306.3.110 `rfind()` [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
 const _CharT * __s,
 size_type __pos = npos) const [inline], [noexcept]
```

Find last position of a C string.

##### Parameters

|                    |                                                      |
|--------------------|------------------------------------------------------|
| <code>__s</code>   | C string to locate.                                  |
| <code>__pos</code> | Index of character to start search at (default end). |



**Returns**

Index of start of last occurrence.

Starting from `__pos`, searches backward for the value of `__s` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5364 of file `basic_string.h`.

**4.306.3.111 rfind()** [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc >
basic_string<_CharT, _Traits, _Alloc >::size_type std::basic_string<_CharT, _Traits, _Alloc
>::rfind (
 _CharT __c,
 size_type __pos = npos) const [noexcept]
```

Find last position of a character.

**Parameters**

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__c</code>   | Character to locate.                                  |
| <code>__pos</code> | Index of character to search back from (default end). |

**Returns**

Index of last occurrence.

Starting from `__pos`, searches backward for `__c` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 1267 of file `basic_string.tcc`.

**4.306.3.112 shrink\_to\_fit()**

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc >::shrink_to_fit () [inline], [noexcept]
```

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 3957 of file `basic_string.h`.

**4.306.3.113** `size()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3913 of file `basic_string.h`.

Referenced by `std::basic_string< char >::append()`, `std::basic_regex< _Ch_type, _Rx_traits >::assign()`, `std::basic_string< char >::assign()`, `std::basic_string< char >::at()`, `std::basic_string< char >::cend()`, `std::basic_string< char >::compare()`, `std::basic_string< char >::empty()`, `std::basic_string< char >::end()`, `std::match_results< _Bi_iter >::format()`, `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes()`, `std::operator+`, `std::basic_string< char >::push_back()`, `std::basic_string< char >::shrink_to_fit()`, `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`, and `std::regex_traits< _CharType >::transform()`.

**4.306.3.114** `substr()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
 size_type __pos = 0,
 size_type __n = npos) const [inline]
```

Get a substring.

**Parameters**

|                    |                                                        |
|--------------------|--------------------------------------------------------|
| <code>__pos</code> | Index of first character (default 0).                  |
| <code>__n</code>   | Number of characters in substring (default remainder). |

**Returns**

The new string.

**Exceptions**

|                                |                                     |
|--------------------------------|-------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos &gt; size()</code> . |
|--------------------------------|-------------------------------------|

Construct and return a new string using the `__n` characters starting at `__pos`. If the string is too short, use the remainder of the characters. If `__pos` is beyond the end of the string, `out_of_range` is thrown.

Definition at line 5744 of file `basic_string.h`.

4.306.3.115 `swap()`

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string<_CharT, _Traits, _Alloc>::swap (
 basic_string<_CharT, _Traits, _Alloc> & __s) [noexcept]
```

Swap contents with another string.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 963 of file `basic_string.tcc`.

Referenced by `std::basic_string<char>::assign()`, and `std::basic_string<char>::operator=()`.

## 4.306.4 Member Data Documentation

4.306.4.1 `npos`

```
template<typename _CharT, typename _Traits, typename _Alloc>
const basic_string<_CharT, _Traits, _Alloc>::size_type std::basic_string<_CharT, _Traits, _
_Alloc>::npos [static]
```

Value returned by various member functions when they fail.

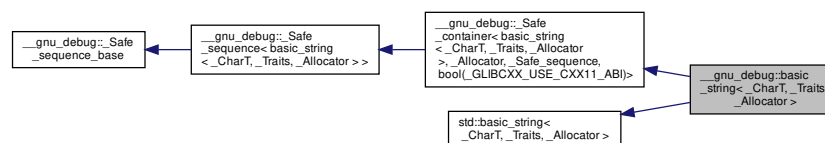
Definition at line 3353 of file `basic_string.h`.

The documentation for this class was generated from the following files:

- [basic\\_string.h](#)
- [basic\\_string.tcc](#)

4.307 `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>` Class Template Reference

Inheritance diagram for `__gnu_debug::basic_string<_CharT, _Traits, _Allocator>`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator`< typename `_Base::const_iterator`, `basic_string` > **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator`< `const_iterator` > **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator`< typename `_Base::iterator`, `basic_string` > **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator`< `iterator` > **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Traits` **traits\_type**
- typedef `_Traits::char_type` **value\_type**

## Public Member Functions

- **basic\_string** (const `_Allocator` &\_\_a) noexcept
- **basic\_string** (const `basic_string` &)=default
- **basic\_string** (`basic_string` &&)=default
- **basic\_string** (`std::initializer_list`< `_CharT` > \_\_l, const `_Allocator` &\_\_a=\_Allocator())
- **basic\_string** (`_Base` &&\_\_base) noexcept
- **basic\_string** (const `_Base` &\_\_base)
- **basic\_string** (const `basic_string` &\_\_str, `size_type` \_\_pos, `size_type` \_\_n=\_Base::npos, const `_Allocator` &\_\_a=\_Allocator())
- **basic\_string** (const `_CharT` \* \_\_s, `size_type` \_\_n, const `_Allocator` &\_\_a=\_Allocator())
- **basic\_string** (const `_CharT` \* \_\_s, const `_Allocator` &\_\_a=\_Allocator())
- **basic\_string** (`size_type` \_\_n, `_CharT` \_\_c, const `_Allocator` &\_\_a=\_Allocator())
- template<typename `_InputIterator` >  
**basic\_string** (`_InputIterator` \_\_begin, `_InputIterator` \_\_end, const `_Allocator` &\_\_a=\_Allocator())
- `_Base` & **\_M\_base** () noexcept
- const `_Base` & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (`_Predicate` \_\_pred)
- `basic_string`< `_CharT`, `_Traits`, `_Allocator` > & **\_M\_replace\_dispatch** (`iterator` \_\_i1, `iterator` \_\_i2, `_InputIterator` \_\_k1, `_InputIterator` \_\_k2, `_false_type`)
- void **\_M\_swap** (`_Safe_container` &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `_CharT` \* **\_S\_construct** (`_InIterator` \_\_beg, `_InIterator` \_\_end, const `_Allocator` &\_\_a, `forward_iterator_tag`)
- `basic_string` & **append** (const `basic_string` &\_\_str)
- `basic_string` & **append** (const `basic_string` &\_\_str, `size_type` \_\_pos, `size_type` \_\_n)
- `basic_string` & **append** (const `_CharT` \* \_\_s, `size_type` \_\_n)
- `basic_string` & **append** (const `_CharT` \* \_\_s)
- `basic_string` & **append** (`size_type` \_\_n, `_CharT` \_\_c)
- template<typename `_InputIterator` >  
`basic_string` & **append** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `basic_string` & **append** (const `basic_string` &\_\_str)
- `basic_string` & **append** (const `basic_string` &\_\_str, `size_type` \_\_pos, `size_type` \_\_n=npos)
- `basic_string` & **append** (`initializer_list`< `_CharT` > \_\_l)
- `basic_string` & **assign** (const `basic_string` &\_\_x)

- `basic_string` & **assign** (`basic_string` &&\_\_x) noexcept(noexcept(std::declval<\_Base &>().assign(std::move(←\_\_x))))
- `basic_string` & **assign** (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n)
- `basic_string` & **assign** (const \_CharT \*\_\_s, size\_type \_\_n)
- `basic_string` & **assign** (const \_CharT \*\_\_s)
- `basic_string` & **assign** (size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator >  
`basic_string` & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `basic_string` & **assign** (std::initializer\_list<\_CharT> \_\_l)
- `basic_string` & **assign** (const `basic_string` &\_\_str)
- `basic_string` & **assign** (`basic_string` &&\_\_str) noexcept(allocator\_traits<\_Allocator>::is\_always\_equal::value)
- `basic_string` & **assign** (const `basic_string` &\_\_str, size\_type \_\_pos, size\_type \_\_n=`npos`)
- const\_reference **at** (size\_type \_\_n) const
- reference **at** (size\_type \_\_n)
- reference **back** ()
- const\_reference **back** () const noexcept
- **iterator begin** ()
- const\_iterator **begin** () const noexcept
- const \_CharT \* **c\_str** () const noexcept
- size\_type **capacity** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** ()
- int **compare** (const `basic_string` &\_\_str) const
- int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str) const
- int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- int **compare** (const \_CharT \*\_\_s) const
- int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_s) const
- int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2) const
- int **compare** (const `basic_string` &\_\_str) const
- int **compare** (size\_type \_\_pos, size\_type \_\_n, const `basic_string` &\_\_str) const
- int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const `basic_string` &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=`npos`) const
- size\_type **copy** (\_CharT \*\_\_s, size\_type \_\_n, size\_type \_\_pos=0) const
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- const \_CharT \* **data** () const noexcept
- bool **empty** () const noexcept
- **iterator end** ()
- const\_iterator **end** () const noexcept
- `basic_string` & **erase** (size\_type \_\_pos=0, size\_type \_\_n=`Base::npos`)
- **iterator erase** (**iterator** \_\_position)
- **iterator erase** (**iterator** \_\_first, **iterator** \_\_last)
- **iterator erase** (**iterator** \_\_position)
- **iterator erase** (**iterator** \_\_first, **iterator** \_\_last)
- size\_type **find** (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type **find** (const \_CharT \*\_\_s, size\_type \_\_pos=0) const
- size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- size\_type **find** (const `basic_string` &\_\_str, size\_type \_\_pos=0) const noexcept

- `size_type find_first_not_of` (const `basic_string` & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type find_first_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- `size_type find_first_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos=0) const
- `size_type find_first_not_of` ( `_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- `size_type find_first_not_of` (const `basic_string` & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type find_first_of` (const `basic_string` & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type find_first_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- `size_type find_first_of` (const `_CharT` \* \_\_s, size\_type \_\_pos=0) const
- `size_type find_first_of` ( `_CharT` \_\_c, size\_type \_\_pos=0) const noexcept
- `size_type find_first_of` (const `basic_string` & \_\_str, size\_type \_\_pos=0) const noexcept
- `size_type find_last_not_of` (const `basic_string` & \_\_str, size\_type \_\_pos= `Base::npos`) const noexcept
- `size_type find_last_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- `size_type find_last_not_of` (const `_CharT` \* \_\_s, size\_type \_\_pos= `Base::npos`) const
- `size_type find_last_not_of` ( `_CharT` \_\_c, size\_type \_\_pos= `Base::npos`) const noexcept
- `size_type find_last_not_of` (const `basic_string` & \_\_str, size\_type \_\_pos= `npos`) const noexcept
- `size_type find_last_of` (const `basic_string` & \_\_str, size\_type \_\_pos= `Base::npos`) const noexcept
- `size_type find_last_of` (const `_CharT` \* \_\_s, size\_type \_\_pos, size\_type \_\_n) const
- `size_type find_last_of` (const `_CharT` \* \_\_s, size\_type \_\_pos= `Base::npos`) const
- `size_type find_last_of` ( `_CharT` \_\_c, size\_type \_\_pos= `Base::npos`) const noexcept
- `size_type find_last_of` (const `basic_string` & \_\_str, size\_type \_\_pos= `npos`) const noexcept
- reference `front` ()
- const\_reference `front` () const noexcept
- allocator\_type `get_allocator` () const noexcept
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` & \_\_str)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` & \_\_str, size\_type \_\_pos2, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const `_CharT` \* \_\_s, size\_type \_\_n)
- `basic_string` & `insert` (size\_type \_\_pos, const `_CharT` \* \_\_s)
- `basic_string` & `insert` (size\_type \_\_pos, size\_type \_\_n, `_CharT` \_\_c)
- `iterator insert` ( `_const_iterator` \_\_p, `_CharT` \_\_c)
- `iterator insert` (const `iterator` \_\_p, size\_type \_\_n, `_CharT` \_\_c)
- `template<typename _InputIterator >`  
`iterator insert` ( `_const_iterator` \_\_p, `_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator insert` (const `iterator` \_\_p, `std::initializer_list`< `_CharT` > \_\_l)
- void `insert` (`iterator` \_\_p, size\_type \_\_n, `_CharT` \_\_c)
- void `insert` (`iterator` \_\_p, `_InputIterator` \_\_beg, `_InputIterator` \_\_end)
- void `insert` (`iterator` \_\_p, `initializer_list`< `_CharT` > \_\_l)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` & \_\_str)
- `basic_string` & `insert` (size\_type \_\_pos1, const `basic_string` & \_\_str, size\_type \_\_pos2, size\_type \_\_n= `npos`)
- `iterator insert` (`iterator` \_\_p, `_CharT` \_\_c)
- `size_type length` () const noexcept
- `size_type max_size` () const noexcept
- `basic_string` & `operator+=` (const `basic_string` & \_\_str)
- `basic_string` & `operator+=` (const `_CharT` \* \_\_s)
- `basic_string` & `operator+=` ( `_CharT` \_\_c)
- `basic_string` & `operator+=` (`std::initializer_list`< `_CharT` > \_\_l)
- `basic_string` & `operator+=` (const `basic_string` & \_\_str)
- `basic_string` & `operator=` (const `basic_string` &)=default
- `basic_string` & `operator=` (`basic_string` &&)=default
- `basic_string` & `operator=` (const `_CharT` \* \_\_s)
- `basic_string` & `operator=` ( `_CharT` \_\_c)
- `basic_string` & `operator=` (`std::initializer_list`< `_CharT` > \_\_l)

- const\_reference **operator[]** (size\_type \_\_pos) const noexcept
- reference **operator[]** (size\_type \_\_pos)
- void **pop\_back** ()
- void **push\_back** (\_CharT \_\_c)
- [reverse\\_iterator](#) **rbegin** ()
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** ()
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- [basic\\_string](#) & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s, size\_type \_\_n2)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, const \_CharT \*\_\_s)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n1, size\_type \_\_n2, \_CharT \_\_c)
- [basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, const \_CharT \*\_\_s)
- [basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- template<typename \_InputIterator >  
[basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, \_InputIterator \_\_j1, \_InputIterator \_\_j2)
- [basic\\_string](#) & **replace** (\_\_const\_iterator \_\_i1, \_\_const\_iterator \_\_i2, std::initializer\_list<\_CharT> \_\_l)
- [basic\\_string](#) & **replace** (size\_type \_\_pos, size\_type \_\_n, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (size\_type \_\_pos1, size\_type \_\_n1, const [basic\\_string](#) &\_\_str, size\_type \_\_pos2, size\_type \_\_n2=npos)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const [basic\\_string](#) &\_\_str)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s, size\_type \_\_n)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_s)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, size\_type \_\_n, \_CharT \_\_c)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, \_InputIterator \_\_k1, \_InputIterator \_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, \_CharT \*\_\_k1, \_CharT \*\_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const \_CharT \*\_\_k1, const \_CharT \*\_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, iterator \_\_k1, iterator \_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, const\_iterator \_\_k1, const\_iterator \_\_k2)
- [basic\\_string](#) & **replace** (iterator \_\_i1, iterator \_\_i2, initializer\_list<\_CharT> \_\_l)
- void **reserve** (size\_type \_\_res\_arg=0)
- void **resize** (size\_type \_\_n, \_CharT \_\_c)
- void **resize** (size\_type \_\_n)
- size\_type **rfind** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=\_Base::npos) const noexcept
- size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos, size\_type \_\_n) const
- size\_type **rfind** (const \_CharT \*\_\_s, size\_type \_\_pos=\_Base::npos) const
- size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=\_Base::npos) const noexcept
- size\_type **rfind** (const [basic\\_string](#) &\_\_str, size\_type \_\_pos=npos) const noexcept
- void **shrink\_to\_fit** () noexcept
- size\_type **size** () const noexcept
- [basic\\_string](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=\_Base::npos) const
- void **swap** ([basic\\_string](#) &\_\_x) noexcept(*/\*conditional \*/*)
- void **swap** ([basic\\_string](#) &\_\_s) noexcept(*/\*conditional \*/*)

### Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

### Static Public Attributes

- static const size\_type [npos](#)

### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) ([\\_Safe\\_sequence\\_base](#) &\_\_x) noexcept

### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
class `::__gnu_debug::Safe_iterator`

#### 4.307.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>>
class __gnu_debug::basic_string<_CharT, _Traits, _Allocator >
```

Class `std::basic_string` with safety/checking/debug instrumentation.

Definition at line 86 of file `debug/string`.

#### 4.307.2 Member Function Documentation

##### 4.307.2.1 [\\_M\\_detach\\_all](#)()

```
void __gnu_debug::Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::Safe_sequence_base::~Safe_sequence_base()`.



## 4.307.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

## 4.307.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 4.307.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

## 4.307.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_invalidate←
_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

#### 4.307.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.307.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.307.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< basic_string< _CharT, _Traits, _Allocator > >::_M_transfer_↵
from_if (
 _Safe_sequence< basic_string< _CharT, _Traits, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

#### 4.307.2.9 `append()` [1/3]

```
basic_string< _CharT, _Traits, _Allocator > & std::basic_string< _CharT, _Traits, _Allocator >↵
::append (
 const basic_string< _CharT, _Traits, _Allocator > & __str) [inherited]
```

Append a string to this string.

##### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

##### Returns

Reference to this string.

Definition at line 769 of file `basic_string.tcc`.

## 4.307.2.10 append() [2/3]

```
basic_string<_CharT, _Traits, _Allocator> & std::basic_string<_CharT, _Traits, _Allocator>::append (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos,
 size_type __n = npos) [inherited]
```

Append a substring.

## Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__str</code> | The string to append.                                       |
| <code>__pos</code> | Index of the first character of <code>str</code> to append. |
| <code>__n</code>   | The number of characters to append.                         |

## Returns

Reference to this string.

## Exceptions

|                                |                                             |
|--------------------------------|---------------------------------------------|
| <code>std::out_of_range</code> | if <code>__pos</code> is not a valid index. |
|--------------------------------|---------------------------------------------|

This function appends `__n` characters from `__str` starting at `__pos` to this string. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is appended.

Definition at line 786 of file `basic_string.tcc`.

## 4.307.2.11 append() [3/3]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::append (
 initializer_list<_CharT> __l) [inline], [inherited]
```

Append an `initializer_list` of characters.

## Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__l</code> | The <code>initializer_list</code> of characters to append. |
|------------------|------------------------------------------------------------|

**Returns**

Reference to this string.

Definition at line 4276 of file basic\_string.h.

**4.307.2.12 assign()** [1/3]

```
basic_string<_CharT, _Traits, _Allocator> & std::basic_string<_CharT, _Traits, _Allocator>::assign (
 const basic_string<_CharT, _Traits, _Allocator> & __str) [inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 687 of file basic\_string.tcc.

**4.307.2.13 assign()** [2/3]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (
 basic_string<_CharT, _Traits, _Allocator> && __str) [inline], [noexcept], [inherited]
```

Set value to contents of another string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | Source string to use. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

This function sets this string to the exact contents of `__str`. `__str` is a valid, but unspecified string.

Definition at line 4358 of file basic\_string.h.

4.307.2.14 `assign()` [3/3]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::assign (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos,
 size_type __n = npos) [inline], [inherited]
```

Set value to a substring of a string.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__str</code> | The string to use.                   |
| <code>__pos</code> | Index of the first character of str. |
| <code>__n</code>   | Number of characters to use.         |

## Returns

Reference to this string.

## Exceptions

|                                |                                           |
|--------------------------------|-------------------------------------------|
| <code>std::out_of_range</code> | if <code>pos</code> is not a valid index. |
|--------------------------------|-------------------------------------------|

This function sets this string to the substring of `__str` consisting of `__n` characters at `__pos`. If `__n` is larger than the number of available characters in `__str`, the remainder of `__str` is used.

Definition at line 4380 of file `basic_string.h`.

4.307.2.15 `at()` [1/2]

```
const_reference std::basic_string<_CharT, _Traits, _Allocator>::at (
 size_type __n) const [inline], [inherited]
```

Provides access to the data contained in the string.

## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__↔</code> | The index of the character to access. |
| <code>__n</code> |                                       |

## Returns

Read-only (const) reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails.

Definition at line 4080 of file `basic_string.h`.

**4.307.2.16** `at()` [2/2]

```
reference std::basic_string<_CharT, _Traits, _Allocator>::at (
 size_type __n) [inline], [inherited]
```

Provides access to the data contained in the string.

**Parameters**

|                          |                                       |
|--------------------------|---------------------------------------|
| <code>__↔<br/>__n</code> | The index of the character to access. |
|--------------------------|---------------------------------------|

**Returns**

Read/write reference to the character.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If $n$ is an invalid index. |
|--------------------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws `out_of_range` if the check fails. Success results in unsharing the string.

Definition at line 4102 of file `basic_string.h`.

**4.307.2.17** `back()` [1/2]

```
reference std::basic_string<_CharT, _Traits, _Allocator>::back () [inline], [inherited]
```

Returns a read/write reference to the data at the last element of the string.

Definition at line 4141 of file `basic_string.h`.

**4.307.2.18 back()** [2/2]

```
const_reference std::basic_string< _CharT, _Traits, _Allocator >::back () const [inline], [noexcept],
[inherited]
```

Returns a read-only (constant) reference to the data at the last element of the string.

Definition at line 4152 of file basic\_string.h.

**4.307.2.19 capacity()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::capacity () const [inline], [noexcept],
[inherited]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Definition at line 3976 of file basic\_string.h.

**4.307.2.20 compare()** [1/3]

```
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
 const basic_string< _CharT, _Traits, _Allocator > & __str) const [inline], [inherited]
```

Compare to a string.

**Parameters**

|                    |                            |
|--------------------|----------------------------|
| <code>__str</code> | String to compare against. |
|--------------------|----------------------------|

**Returns**

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before `__str`, 0 if their values are equivalent, or > 0 if this string is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of `size()` and `str.size()`. The function then compares the two strings by calling `traits::compare(data(), str.data(), rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 5763 of file basic\_string.h.

**4.307.2.21 compare()** [2/3]

```
int std::basic_string<_CharT, _Traits, _Allocator >::compare (
 size_type __pos,
 size_type __n,
 const basic_string<_CharT, _Traits, _Allocator > & __str) const [inherited]
```

Compare substring to a string.



## Parameters

|                    |                                        |
|--------------------|----------------------------------------|
| <code>__pos</code> | Index of first character of substring. |
| <code>__n</code>   | Number of characters in substring.     |
| <code>__str</code> | String to compare against.             |

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n` characters starting at `__pos`. Returns an integer  $< 0$  if the substring is ordered before `__str`,  $0$  if their values are equivalent, or  $> 0$  if the substring is ordered after `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the length of the substring and `__str.size()`. The function then compares the two strings by calling `traits::compare(substring.data(),str.data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1388 of file `basic_string.tcc`.

4.307.2.22 `compare()` [3/3]

```
int std::basic_string<_CharT, _Traits, _Allocator>::compare (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos2,
 size_type __n2 = npos) const [inherited]
```

Compare substring to a substring.

## Parameters

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| <code>__pos1</code> | Index of first character of substring.                      |
| <code>__n1</code>   | Number of characters in substring.                          |
| <code>__str</code>  | String to compare against.                                  |
| <code>__pos2</code> | Index of first character of substring of <code>str</code> . |
| <code>__n2</code>   | Number of characters in substring of <code>str</code> .     |

## Returns

Integer  $< 0$ ,  $0$ , or  $> 0$ .

Form the substring of this string from the `__n1` characters starting at `__pos1`. Form the substring of `__str` from the `__n2` characters starting at `__pos2`. Returns an integer  $< 0$  if this substring is ordered before the substring of `__str`,  $0$  if their values are equivalent, or  $> 0$  if this substring is ordered after the substring of `__str`. Determines the effective length `rlen` of the strings to compare as the smallest of the lengths of the substrings. The function then compares the

two strings by calling `traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen)`. If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Definition at line 1403 of file `basic_string.tcc`.

#### 4.307.2.23 `empty()`

```
bool std::basic_string<_CharT, _Traits, _Allocator >::empty () const [inline], [noexcept],
[inherited]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

Definition at line 4026 of file `basic_string.h`.

#### 4.307.2.24 `erase()` [1/2]

```
iterator std::basic_string<_CharT, _Traits, _Allocator >::erase (
 iterator __position) [inline], [inherited]
```

Remove one character.

##### Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | Iterator referencing the character to remove. |
|-------------------------|-----------------------------------------------|

##### Returns

iterator referencing same location after removal.

Removes the character at `__position` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 4726 of file `basic_string.h`.

#### 4.307.2.25 `erase()` [2/2]

```
basic_string<_CharT, _Traits, _Allocator >::iterator std::basic_string<_CharT, _Traits, _↵
Allocator >::erase (
 iterator __first,
 iterator __last) [inherited]
```

Remove a range of characters.

## Parameters

|                      |                                                     |
|----------------------|-----------------------------------------------------|
| <code>__first</code> | Iterator referencing the first character to remove. |
| <code>__last</code>  | Iterator referencing the end of the range.          |

## Returns

Iterator referencing location of first after removal.

Removes the characters in the range `[first,last)` from this string. The value of the string doesn't change if an error is thrown.

Definition at line 835 of file `basic_string.tcc`.

4.307.2.26 `find()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = 0) const [inline], [noexcept], [inherited]
```

Find position of a string.

## Parameters

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String to locate.                              |
| <code>__pos</code> | Index of character to search from (default 0). |

## Returns

Index of start of first occurrence.

Starting from `__pos`, searches forward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5271 of file `basic_string.h`.

4.307.2.27 `find_first_not_of()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_first_not_of (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = 0) const [inline], [noexcept], [inherited]
```

Find position of a character not in string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.         |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5579 of file `basic_string.h`.

**4.307.2.28 find\_first\_of()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = 0) const [inline], [noexcept], [inherited]
```

Find position of a character of string.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__str</code> | String containing characters to locate.        |
| <code>__pos</code> | Index of character to search from (default 0). |

**Returns**

Index of first occurrence.

Starting from `__pos`, searches forward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5412 of file `basic_string.h`.

**4.307.2.29 find\_last\_not\_of()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
 const basic_string< _CharT, _Traits, _Allocator > & __str,
 size_type __pos = npos) const [inline], [noexcept], [inherited]
```

Find last position of a character not in string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to avoid.                |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for a character not contained in `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5661 of file `basic_string.h`.

4.307.2.30 `find_last_of()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::find_last_of (
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos = npos) const [inline], [noexcept], [inherited]
```

Find last position of a character of string.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String containing characters to locate.               |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of last occurrence.

Starting from `__pos`, searches backward for one of the characters of `__str` within this string. If found, returns the index where it was found. If not found, returns `npos`.

Definition at line 5496 of file `basic_string.h`.

4.307.2.31 `front()` [1/2]

```
reference std::basic_string<_CharT, _Traits, _Allocator>::front () [inline], [inherited]
```

Returns a read/write reference to the data at the first element of the string.

Definition at line 4119 of file `basic_string.h`.

**4.307.2.32** `front()` [2/2]

```
const_reference std::basic_string< _CharT, _Traits, _Allocator >::front () const [inline],
[noexcept], [inherited]
```

Returns a read-only (constant) reference to the data at the first element of the string.

Definition at line 4130 of file `basic_string.h`.

**4.307.2.33** `get_allocator()`

```
allocator_type std::basic_string< _CharT, _Traits, _Allocator >::get_allocator () const [inline],
[noexcept], [inherited]
```

Return copy of allocator used to construct this string.

Definition at line 5241 of file `basic_string.h`.

**4.307.2.34** `insert()` [1/6]

```
void std::basic_string< _CharT, _Traits, _Allocator >::insert (
 iterator __p,
 size_type __n,
 _CharT __c) [inline], [inherited]
```

Insert multiple characters.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at. |
| <code>__n</code> | Number of characters to insert                        |
| <code>__c</code> | The character to insert.                              |

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts `__n` copies of character `__c` starting at the position referenced by iterator `__p`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4497 of file `basic_string.h`.

4.307.2.35 `insert()` [2/6]

```
void std::basic_string<_CharT, _Traits, _Allocator>::insert (
 iterator __p,
 _InputIterator __beg,
 _InputIterator __end) [inline], [inherited]
```

Insert a range of characters.

## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__p</code>   | Iterator referencing location in string to insert at. |
| <code>__beg</code> | Start of range.                                       |
| <code>__end</code> | End of range.                                         |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts characters in range `[__beg, __end)`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4514 of file `basic_string.h`.

4.307.2.36 `insert()` [3/6]

```
void std::basic_string<_CharT, _Traits, _Allocator>::insert (
 iterator __p,
 initializer_list<_CharT> __l) [inline], [inherited]
```

Insert an `initializer_list` of characters.

## Parameters

|                  |                                                            |
|------------------|------------------------------------------------------------|
| <code>__p</code> | Iterator referencing location in string to insert at.      |
| <code>__l</code> | The <code>initializer_list</code> of characters to insert. |

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Definition at line 4525 of file `basic_string.h`.

**4.307.2.37** `insert()` [4/6]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Allocator> & __str) [inline], [inherited]
```

Insert value of a string.

**Parameters**

|                     |                                  |
|---------------------|----------------------------------|
| <code>__pos1</code> | Position in string to insert at. |
| <code>__str</code>  | The string to insert.            |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts value of `__str` starting at `__pos1`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4545 of file `basic_string.h`.

**4.307.2.38** `insert()` [5/6]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::insert (
 size_type __pos1,
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos2,
 size_type __n = npos) [inline], [inherited]
```

Insert a substring.

**Parameters**

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <code>__pos1</code> | Position in string to insert at.                   |
| <code>__str</code>  | The string to insert.                              |
| <code>__pos2</code> | Start of characters in <code>str</code> to insert. |
| <code>__n</code>    | Number of characters to insert.                    |



**Returns**

Reference to this string.

**Exceptions**

|                                |                                                                           |
|--------------------------------|---------------------------------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                           |
| <code>std::out_of_range</code> | If <code>pos1 &gt; size()</code> or <code>__pos2 &gt; str.size()</code> . |

Starting at `pos1`, insert `__n` character of `__str` beginning with `__pos2`. If adding characters causes the length to exceed `max_size()`, `length_error` is thrown. If `__pos1` is beyond the end of this string or `__pos2` is beyond the end of `__str`, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4567 of file `basic_string.h`.

**4.307.2.39 `insert()`** [6/6]

```
iterator std::basic_string<_CharT, _Traits, _Allocator>::insert (
 iterator __p,
 _CharT __c) [inline], [inherited]
```

Insert one character.

**Parameters**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <code>__p</code> | Iterator referencing position in string to insert at. |
| <code>__c</code> | The character to insert.                              |

**Returns**

Iterator referencing newly inserted char.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Inserts character `__c` at position referenced by `__p`. If adding character causes the length to exceed `max_size()`, `length_error` is thrown. If `__p` is beyond end of string, `out_of_range` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4649 of file `basic_string.h`.

**4.307.2.40 length()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::length () const [inline], [noexcept],
[inherited]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3919 of file basic\_string.h.

**4.307.2.41 max\_size()**

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::max_size () const [inline], [noexcept],
[inherited]
```

Returns the size() of the largest possible string.

Definition at line 3924 of file basic\_string.h.

**4.307.2.42 operator+=( )**

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::operator+= (
 const basic_string< _CharT, _Traits, _Allocator > & __str) [inline], [inherited]
```

Append a string to this string.

**Parameters**

|                    |                       |
|--------------------|-----------------------|
| <code>__str</code> | The string to append. |
|--------------------|-----------------------|

**Returns**

Reference to this string.

Definition at line 4166 of file basic\_string.h.

**4.307.2.43 replace()** [1/8]

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (
 size_type __pos,
 size_type __n,
 const basic_string< _CharT, _Traits, _Allocator > & __str) [inline], [inherited]
```

Replace characters with value from another string.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__pos</code> | Index of first character to replace. |
| <code>__n</code>   | Number of characters to be replaced. |
| <code>__str</code> | String to insert.                    |

## Returns

Reference to this string.

## Exceptions

|                                |                                                       |
|--------------------------------|-------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is beyond the end of this string. |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .       |

Removes the characters in the range `[__pos, __pos+__n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4780 of file `basic_string.h`.

4.307.2.44 `replace()` [2/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
 size_type __pos1,
 size_type __n1,
 const basic_string<_CharT, _Traits, _Allocator> & __str,
 size_type __pos2,
 size_type __n2 = npos) [inline], [inherited]
```

Replace characters with value from another string.

## Parameters

|                     |                                                      |
|---------------------|------------------------------------------------------|
| <code>__pos1</code> | Index of first character to replace.                 |
| <code>__n1</code>   | Number of characters to be replaced.                 |
| <code>__str</code>  | String to insert.                                    |
| <code>__pos2</code> | Index of first character of <code>str</code> to use. |
| <code>__n2</code>   | Number of characters from <code>str</code> to use.   |

## Returns

Reference to this string.

**Exceptions**

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos1 &gt; size()</code> or <code>__pos2 &gt; __str.size()</code> . |
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> .                               |

Removes the characters in the range `[__pos1, __pos1 + n)` from this string. In place, the value of `__str` is inserted. If `__pos` is beyond end of string, `out_of_range` is thrown. If the length of the result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4802 of file `basic_string.h`.

**4.307.2.45 `replace()`** [3/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
 iterator __i1,
 iterator __i2,
 const basic_string<_CharT, _Traits, _Allocator> & __str) [inline], [inherited]
```

Replace range of characters with string.

**Parameters**

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__i1</code>  | Iterator referencing start of range to replace. |
| <code>__i2</code>  | Iterator referencing end of range to replace.   |
| <code>__str</code> | String value to insert.                         |

**Returns**

Reference to this string.

**Exceptions**

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the value of `__str` is inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4889 of file `basic_string.h`.

**4.307.2.46 `replace()`** [4/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
 iterator __i1,
```

```

 iterator __i2,
 const _CharT * __s,
 size_type __n) [inline], [inherited]

```

Replace range of characters with C substring.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |
| <code>__n</code>  | Number of characters from s to insert.          |

#### Returns

Reference to this string.

#### Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1, __i2)`. In place, the first `__n` characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4908 of file `basic_string.h`.

#### 4.307.2.47 `replace()` [5/8]

```

basic_string& std::basic_string<_CharT, _Traits, _Allocator>::replace (
 iterator __i1,
 iterator __i2,
 const _CharT * __s) [inline], [inherited]

```

Replace range of characters with C string.

#### Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__s</code>  | C string value to insert.                       |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, the characters of `__s` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4929 of file `basic_string.h`.

**4.307.2.48 replace()** [6/8]

```
basic_string& std::basic_string<_CharT, _Traits, _Allocator >::replace (
 iterator __i1,
 iterator __i2,
 size_type __n,
 _CharT __c) [inline], [inherited]
```

Replace range of characters with multiple characters.

**Parameters**

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__n</code>  | Number of characters to insert.                 |
| <code>__c</code>  | Character to insert.                            |

**Returns**

Reference to this string.

**Exceptions**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| <i>std::length_error</i> | If new length exceeds <code>max_size()</code> . |
|--------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, `__n` copies of `__c` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4950 of file `basic_string.h`.

## 4.307.2.49 replace() [7/8]

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (
 iterator __i1,
 iterator __i2,
 _InputIterator __k1,
 _InputIterator __k2) [inline], [inherited]
```

Replace range of characters with range.

## Parameters

|                   |                                                 |
|-------------------|-------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace. |
| <code>__i2</code> | Iterator referencing end of range to replace.   |
| <code>__k1</code> | Iterator referencing start of range to insert.  |
| <code>__k2</code> | Iterator referencing end of range to insert.    |

## Returns

Reference to this string.

## Exceptions

|                                |                                                 |
|--------------------------------|-------------------------------------------------|
| <code>std::length_error</code> | If new length exceeds <code>max_size()</code> . |
|--------------------------------|-------------------------------------------------|

Removes the characters in the range [`__i1`,`__i2`). In place, characters in the range [`__k1`,`__k2`) are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 4974 of file `basic_string.h`.

## 4.307.2.50 replace() [8/8]

```
basic_string& std::basic_string< _CharT, _Traits, _Allocator >::replace (
 iterator __i1,
 iterator __i2,
 initializer_list< _CharT > __l) [inline], [inherited]
```

Replace range of characters with `initializer_list`.

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <code>__i1</code> | Iterator referencing start of range to replace.            |
| <code>__i2</code> | Iterator referencing end of range to replace.              |
| <code>__l</code>  | The <code>initializer_list</code> of characters to insert. |

**Returns**

Reference to this string.

**Exceptions**

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <code><i>std::length_error</i></code> | If new length exceeds <code>max_size()</code> . |
|---------------------------------------|-------------------------------------------------|

Removes the characters in the range `[__i1,__i2)`. In place, characters in the range `[__k1,__k2)` are inserted. If the length of result exceeds `max_size()`, `length_error` is thrown. The value of the string doesn't change if an error is thrown.

Definition at line 5043 of file `basic_string.h`.

**4.307.2.51 `reserve()`**

```
void std::basic_string<_CharT, _Traits, _Allocator >::reserve (
 size_type __res_arg = 0) [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

**Parameters**

|                        |                                |
|------------------------|--------------------------------|
| <code>__res_arg</code> | Number of characters required. |
|------------------------|--------------------------------|

**Exceptions**

|                                       |                                                             |
|---------------------------------------|-------------------------------------------------------------|
| <code><i>std::length_error</i></code> | If <code>__res_arg</code> exceeds <code>max_size()</code> . |
|---------------------------------------|-------------------------------------------------------------|

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

Definition at line 946 of file `basic_string.tcc`.

**4.307.2.52 `rfind()`**

```
size_type std::basic_string<_CharT, _Traits, _Allocator >::rfind (
 const basic_string<_CharT, _Traits, _Allocator > & __str,
 size_type __pos = npos) const [inline], [noexcept], [inherited]
```

Find last position of a string.



## Parameters

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| <code>__str</code> | String to locate.                                     |
| <code>__pos</code> | Index of character to search back from (default end). |

## Returns

Index of start of last occurrence.

Starting from `__pos`, searches backward for value of `__str` within this string. If found, returns the index where it begins. If not found, returns `npos`.

Definition at line 5333 of file `basic_string.h`.

4.307.2.53 `size()`

```
size_type std::basic_string<_CharT, _Traits, _Allocator>::size () const [inline], [noexcept],
[inherited]
```

Returns the number of characters in the string, not including any null-termination.

Definition at line 3913 of file `basic_string.h`.

4.307.2.54 `swap()`

```
void std::basic_string<_CharT, _Traits, _Allocator>::swap (
 basic_string<_CharT, _Traits, _Allocator> & __s) [noexcept], [inherited]
```

Swap contents with another string.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__s</code> | String to swap with. |
|------------------|----------------------|

Exchanges the contents of this string with that of `__s` in constant time.

Definition at line 963 of file `basic_string.tcc`.

## 4.307.3 Member Data Documentation

#### 4.307.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.307.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.307.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

#### 4.307.3.4 `npos`

```
const basic_string< _CharT, _Traits, _Allocator >::size_type std::basic_string< _CharT, _Traits, _Allocator >::npos [static], [inherited]
```

Value returned by various member functions when they fail.

Definition at line 3353 of file `basic_string.h`.

The documentation for this class was generated from the following file:

- [debug/string](#)

## 4.308 std::experimental::fundamentals\_v1::basic\_string\_view&lt; \_CharT, \_Traits &gt; Class Template Reference

## Public Types

- using **const\_iterator** = const \_CharT \*
- using **const\_pointer** = const \_CharT \*
- using **const\_reference** = const \_CharT &
- using **const\_reverse\_iterator** = [std::reverse\\_iterator](#)< const\_iterator >
- using **difference\_type** = ptrdiff\_t
- using **iterator** = const\_iterator
- using **pointer** = \_CharT \*
- using **reference** = \_CharT &
- using **reverse\_iterator** = [const\\_reverse\\_iterator](#)
- using **size\_type** = size\_t
- using **traits\_type** = \_Traits
- using **value\_type** = \_CharT

## Public Member Functions

- constexpr **basic\_string\_view** (const [basic\\_string\\_view](#) &) noexcept=default
- template<typename \_Allocator >  
**basic\_string\_view** (const [basic\\_string](#)< \_CharT, \_Traits, \_Allocator > &\_\_str) noexcept
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str)
- constexpr **basic\_string\_view** (const \_CharT \*\_\_str, size\_type \_\_len)
- constexpr const \_CharT & **at** (size\_type \_\_pos) const
- constexpr const \_CharT & **back** () const
- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **cbegin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr int **compare** ([basic\\_string\\_view](#) \_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, [basic\\_string\\_view](#) \_\_str, size\_type \_\_pos2, size\_type \_\_n2) const
- constexpr int **compare** (const \_CharT \*\_\_str) const noexcept
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str) const
- constexpr int **compare** (size\_type \_\_pos1, size\_type \_\_n1, const \_CharT \*\_\_str, size\_type \_\_n2) const
- size\_type **copy** (\_CharT \*\_\_str, size\_type \_\_n, size\_type \_\_pos=0) const
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- constexpr const \_CharT \* **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **find** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **find** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_not\_of** (const \_CharT \*\_\_str, size\_type \_\_pos=0) const noexcept

- constexpr size\_type **find\_first\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (\_CharT \_\_c, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_first\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=0) const noexcept
- constexpr size\_type **find\_last\_not\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_not\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const
- constexpr size\_type **find\_last\_of** (const \_CharT \* \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr const \_CharT & **front** () const
- constexpr size\_type **length** () const noexcept
- constexpr size\_type **max\_size** () const noexcept
- template<typename \_Allocator >  
**operator basic\_string**<\_CharT, \_Traits, \_Allocator > () const
- [basic\\_string\\_view](#) & **operator=** (const [basic\\_string\\_view](#) &) noexcept=default
- constexpr const \_CharT & **operator[]** (size\_type \_\_pos) const
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- constexpr void **remove\_prefix** (size\_type \_\_n)
- constexpr void **remove\_suffix** (size\_type \_\_n)
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- constexpr size\_type **rfind** ([basic\\_string\\_view](#) \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **rfind** (\_CharT \_\_c, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos, size\_type \_\_n) const noexcept
- constexpr size\_type **rfind** (const \_CharT \* \_\_str, size\_type \_\_pos=npo) const noexcept
- constexpr size\_type **size** () const noexcept
- constexpr [basic\\_string\\_view](#) **substr** (size\_type \_\_pos=0, size\_type \_\_n=npo) const
- constexpr void **swap** ([basic\\_string\\_view](#) & \_\_sv) noexcept
- template<typename \_Allocator = std::allocator<\_CharT>>  
[basic\\_string](#)<\_CharT, \_Traits, \_Allocator > **to\_string** (const \_Allocator & \_\_alloc=\_Allocator()) const

#### Static Public Attributes

- static constexpr size\_type **npo**

#### 4.308.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view<_CharT, _Traits >
```

A non-owning reference to a string.

#### Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character                                                               |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |

A basic\_string\_view looks like this:

```
_CharT* _M_str
size_t _M_len
```

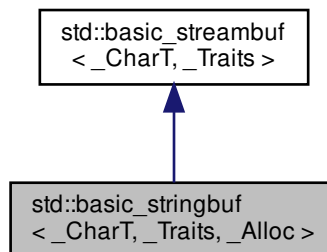
Definition at line 75 of file experimental/string\_view.

The documentation for this class was generated from the following files:

- [experimental/string\\_view](#)
- [experimental/bits/string\\_view.tcc](#)

## 4.309 std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc > Class Template Reference

Inheritance diagram for std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >:



### Public Types

- typedef \_\_string\_type::size\_type **\_\_size\_type**
- typedef [basic\\_streambuf](#)< char\_type, traits\_type > **\_\_streambuf\_type**
- typedef [basic\\_string](#)< char\_type, \_Traits, \_Alloc > **\_\_string\_type**
- typedef \_Alloc **allocator\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef \_Traits **traits\_type**

## Public Member Functions

- [basic\\_stringbuf](#) ()
  - [basic\\_stringbuf](#) ([ios\\_base::openmode](#) \_\_mode)
  - [basic\\_stringbuf](#) (const [\\_\\_string\\_type](#) &\_\_str, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - **basic\_stringbuf** (const [basic\\_stringbuf](#) &)=delete
  - **basic\_stringbuf** ([basic\\_stringbuf](#) &&\_\_rhs)
  - [locale](#) [getloc](#) () const
  - [streamsize](#) [in\\_avail](#) ()
  - [basic\\_stringbuf](#) & **operator=** (const [basic\\_stringbuf](#) &)=delete
  - [basic\\_stringbuf](#) & **operator=** ([basic\\_stringbuf](#) &&\_\_rhs)
  - [locale](#) [pubimbue](#) (const [locale](#) &\_\_loc)
  - int\_type [sbumpc](#) ()
  - int\_type [sgetc](#) ()
  - [streamsize](#) [sgetn](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - int\_type [snextc](#) ()
  - int\_type [sputbackc](#) (char\_type \_\_c)
  - int\_type [sputc](#) (char\_type \_\_c)
  - [streamsize](#) [sputn](#) (const char\_type \*\_\_s, [streamsize](#) \_\_n)
  - [\\_\\_string\\_type](#) [str](#) () const
  - void [str](#) (const [\\_\\_string\\_type](#) &\_\_s)
  - int\_type [sungetc](#) ()
  - void **swap** ([basic\\_stringbuf](#) &\_\_rhs)
- 
- [basic\\_streambuf](#) \* [pubsetbuf](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
  - pos\_type [pubseekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - pos\_type [pubseekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - int [pubsync](#) ()

## Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [\\_M\\_pbump](#) (char\_type \*\_\_pbeg, char\_type \*\_\_pend, off\_type \_\_off)
- void [\\_M\\_stringbuf\\_init](#) ([ios\\_base::openmode](#) \_\_mode)
- void [\\_M\\_sync](#) (char\_type \*\_\_base, \_\_size\_type \_\_i, \_\_size\_type \_\_o)
- void [\\_M\\_update\\_egptr](#) ()
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) &\_\_loc)
- virtual int\_type [overflow](#) (int\_type \_\_c=traits\_type::eof())
- virtual int\_type [pbackfail](#) (int\_type \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual pos\_type [seekoff](#) (off\_type \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual pos\_type [seekpos](#) (pos\_type \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [\\_\\_streambuf\\_type](#) \* [setbuf](#) (char\_type \*\_\_s, [streamsize](#) \_\_n)
- void [setg](#) (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
- void [setp](#) (char\_type \*\_\_pbeg, char\_type \*\_\_pend)

- virtual [streamsize showmanyc](#) ()
- void **swap** ([basic\\_streambuf](#) &\_\_sb)
- virtual int [sync](#) ()
- virtual int\_type [uflow](#) ()
- virtual int\_type [underflow](#) ()
- virtual [streamsize xsgetn](#) (char\_type \* \_\_s, [streamsize](#) \_\_n)
- virtual [streamsize xsputn](#) (const char\_type \* \_\_s, [streamsize](#) \_\_n)

- char\_type \* [eback](#) () const
- char\_type \* [gptr](#) () const
- char\_type \* [egptr](#) () const

- char\_type \* [pbase](#) () const
- char\_type \* [pptr](#) () const
- char\_type \* [epptr](#) () const

#### Protected Attributes

- [locale \\_M\\_buf\\_locale](#)
- char\_type \* [\\_M\\_in\\_beg](#)
- char\_type \* [\\_M\\_in\\_cur](#)
- char\_type \* [\\_M\\_in\\_end](#)
- [ios\\_base::openmode \\_M\\_mode](#)
- char\_type \* [\\_M\\_out\\_beg](#)
- char\_type \* [\\_M\\_out\\_cur](#)
- char\_type \* [\\_M\\_out\\_end](#)
- [\\_\\_string\\_type \\_M\\_string](#)

#### 4.309.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >
```

The actual work of input and output (for std::string).

#### Template Parameters

|                         |                                                                                    |
|-------------------------|------------------------------------------------------------------------------------|
| <a href="#">_CharT</a>  | Type of character stream.                                                          |
| <a href="#">_Traits</a> | Traits for character type, defaults to <a href="#">char_traits&lt;_CharT&gt;</a> . |
| <a href="#">_Alloc</a>  | Allocator type, defaults to <a href="#">allocator&lt;_CharT&gt;</a> .              |

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)

For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

Definition at line 96 of file `iosfwd`.

#### 4.309.2 Constructor & Destructor Documentation

##### 4.309.2.1 `basic_stringbuf()` [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf () [inline]
```

Starts with an empty string buffer.

The default constructor initializes the parent class using its own default ctor.

Definition at line 99 of file `sstream`.

##### 4.309.2.2 `basic_stringbuf()` [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
 ios_base::openmode __mode) [inline], [explicit]
```

Starts with an empty string buffer.

###### Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__mode</code> | Whether the buffer can read, or write, or both. |
|---------------------|-------------------------------------------------|

The default constructor initializes the parent class using its own default ctor.

Definition at line 111 of file `sstream`.

##### 4.309.2.3 `basic_stringbuf()` [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
 const __string_type & __str,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [explicit]
```

Starts with an existing string buffer.



## Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>__str</code>  | A string to copy as a starting buffer.          |
| <code>__mode</code> | Whether the buffer can read, or write, or both. |

This constructor initializes the parent class using its own default ctor.

Definition at line 124 of file sstream.

## 4.309.3 Member Function Documentation

## 4.309.3.1 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 489 of file streambuf.

## 4.309.3.2 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 4.309.3.3 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 4.309.3.4 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

##### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

#### 4.309.3.5 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

##### Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent loc is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

## 4.309.3.6 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

## 4.309.3.7 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, and `std::basic_filebuf< char_type, traits_type >`.

Definition at line 583 of file streambuf.

#### 4.309.3.8 in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

##### Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

#### 4.309.3.9 overflow()

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::overflow (
 int_type __c = traits_type::eof()) [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

##### Parameters

|                 |                                     |
|-----------------|-------------------------------------|
| <code>_↵</code> | An additional character to consume. |
| <code>_c</code> |                                     |

##### Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output `streambuf` can be created by overriding only this function (no buffer area will be used).

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 80 of file `sstream.tcc`.

References `__gnu_debug::__base()`, `std::ios_base::in`, `std::max()`, `std::min()`, `std::ios_base::out`, and `std::basic_string< _CharT, _Traits, _Alloc >::reserve()`.

## 4.309.3.10 pbackfail()

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::pbackfail (
 int_type __c = traits_type::eof()) [protected], [virtual]
```

Tries to back up the input sequence.

## Parameters

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_↵</code> | The character to be inserted back into the sequence. |
| <code>_C</code> |                                                      |

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 46 of file `sstream.tcc`.

References `std::ios_base::out`.

## 4.309.3.11 pbase()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

## 4.309.3.12 pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**4.309.3.13 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**4.309.3.14 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 4.309.3.15 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_stringbuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 4.309.3.16 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_stringbuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

## 4.309.3.17 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_stringbuf* std::basic_stringbuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 4.309.3.18 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 4.309.3.19 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.309.3.20 `seekoff()`

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 168 of file `sstream.tcc`.

References `std::ios_base::cur`, `std::ios_base::end`, `std::ios_base::in`, and `std::ios_base::out`.



## 4.309.3.21 seekpos()

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↵
Alloc >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 216 of file `sstream.tcc`.

References `std::ios_base::in`, and `std::ios_base::out`.

## 4.309.3.22 setbuf()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type* std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
 char_type * __s,
 streamsize __n) [inline], [protected], [virtual]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no buffer has already been created, and both `__s` and `__n` are non-zero, then `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.↵buffering> for more.

Reimplemented from [std::basic\\_streambuf< \\_CharT, \\_Traits >](#).

Definition at line 255 of file `sstream`.

#### 4.309.3.23 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

##### Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

##### Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

#### 4.309.3.24 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

##### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

##### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

#### 4.309.3.25 sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

#### 4.309.3.26 sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

##### Parameters

|       |                |
|-------|----------------|
| $\_s$ | A buffer area. |
| $\_n$ | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

#### 4.309.3.27 showmanyc()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc () [inline],
[protected], [virtual]
```

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 223 of file `sstream`.

#### 4.309.3.28 `snextc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

#### 4.309.3.29 `sputbackc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.309.3.30 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(←__c)`.

Definition at line 431 of file `streambuf`.



Referenced by `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**4.309.3.31 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                                                                                                                       |                     |
|-----------------------------------------------------------------------------------------------------------------------|---------------------|
| <a href="#"></a><br><code>__s</code> | A buffer read area. |
| <a href="#"></a><br><code>__n</code> | A count.            |

One of two public output functions.

Returns `xsputn(__s,__n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**4.309.3.32 `str()`** [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str () const [inline]
```

Copying out the string buffer.

**Returns**

A copy of one of the underlying sequences.

*If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence.* [27.7.1.2]/1


Definition at line 178 of file `sstream`.

**4.309.3.33 `str()`** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

**Parameters**

|                                                                                                                         |                                      |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| <a href="#"></a><br><code>__s</code> | The string to use as a new sequence. |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------|

Deallocates any previous stored sequence, then copies `s` to use as a new one.

Definition at line 202 of file `sstream`.

#### 4.309.3.34 sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

##### Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 404 of file streambuf.

#### 4.309.3.35 sync()

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync (
 void) [inline], [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

##### Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

##### Note

Base class version does nothing, returns zero.

Reimplemented in [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >](#), [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_type, traits\\_type >](#), [std::basic\\_filebuf< char\\_type, traits\\_type >](#), and [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >](#).

Definition at line 634 of file streambuf.

#### 4.309.3.36 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

#### 4.309.3.37 underflow()

```
template<class _CharT , class _Traits , class _Alloc >
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↵
_Alloc >::underflow () [protected], [virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 150 of file `sstream.tcc`.

References `std::ios_base::in`.



#### 4.309.3.38 xsggetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf< _CharT, _Traits >::xsggetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.



## Parameters

|                                                                                                                    |                                         |
|--------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| <a href="#"></a> <code>__s</code> | A buffer area.                          |
| <a href="#"></a> <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).



Definition at line 46 of file `streambuf.tcc`.

## 4.309.3.39 xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                                                                                                                      |                                        |
|----------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a> <code>__s</code> | A buffer area.                         |
| <a href="#"></a> <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf< \\_CharT, \\_Traits >](#), [std::basic\\_filebuf< \\_CharT, encoding\\_char\\_traits< \\_CharT > >](#), and [std::basic\\_filebuf< char\\_type, traits\\_type >](#).

Definition at line 80 of file `streambuf.tcc`.

#### 4.309.4 Member Data Documentation

##### 4.309.4.1 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file streambuf.

##### 4.309.4.2 `_M_in_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

##### 4.309.4.3 `_M_in_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

##### 4.309.4.4 `_M_in_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 4.309.4.5 \_M\_mode

```
template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode [protected]
```

Place to stash in || out || in | out settings for current stringbuf.

Definition at line 85 of file sstream.

#### 4.309.4.6 \_M\_out\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.309.4.7 \_M\_out\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.309.4.8 \_M\_out\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

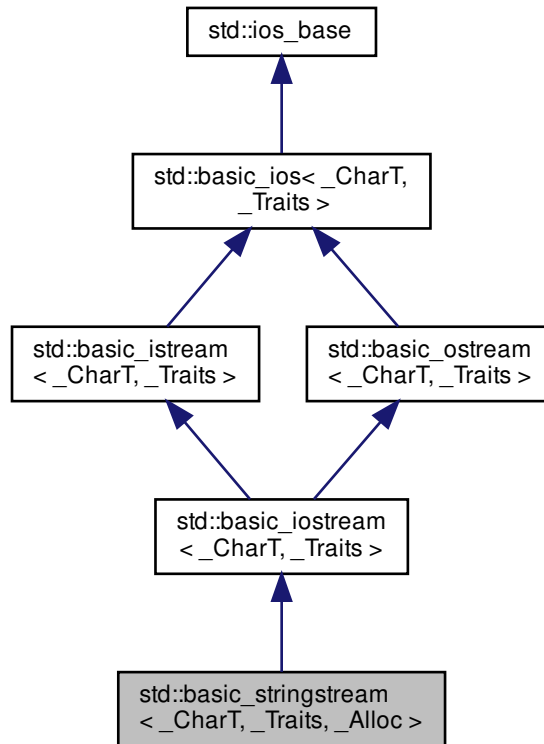
Definition at line 196 of file streambuf.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)
- [sstream.tcc](#)

#### 4.310 `std::basic_stringstream<_CharT, _Traits, _Alloc>` Class Template Reference

Inheritance diagram for `std::basic_stringstream<_CharT, _Traits, _Alloc>`:



#### Public Types

- typedef `ctype<_CharT>` `__ctype_type`
- typedef `ctype<_CharT>` `__ctype_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_ios<_CharT, _Traits>` `__ios_type`
- typedef `basic_istream<char_type, traits_type>` `__istream_type`
- typedef `basic_istream<_CharT, _Traits>` `__istream_type`
- typedef `num_get<_CharT, istreambuf_iterator<_CharT, _Traits>>` `__num_get_type`
- typedef `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`
- typedef `basic_ostream<_CharT, _Traits>` `__ostream_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_streambuf<_CharT, _Traits>` `__streambuf_type`
- typedef `basic_string<_CharT, _Traits, _Alloc>` `__string_type`
- typedef `basic_stringbuf<_CharT, _Traits, _Alloc>` `__stringbuf_type`

- `typedef int io_state` `_GLIBCXX_DEPRECATED_SUGGEST("std::iosstate")`
  - `typedef int open_mode` `_GLIBCXX_DEPRECATED_SUGGEST("std::openmode")`
  - `typedef int seek_dir` `_GLIBCXX_DEPRECATED_SUGGEST("std::seekdir")`
  - `typedef` `std::streampos` `streampos` `_GLIBCXX_DEPRECATED_SUGGEST("std::streampos")`
  - `typedef` `std::streamoff` `streamoff` `_GLIBCXX_DEPRECATED_SUGGEST("std::streamoff")`
  - `typedef` `_Alloc` `allocator_type`
  - `typedef` `_CharT` `char_type`
  - `enum` `event` { `erase_event`, `imbue_event`, `copyfmt_event` }
  - `typedef` `void(* event_callback)` (`event` `__e`, `ios_base` `&__b`, `int` `__i`)
  - `typedef` `_ios_Fmtflags` `fmtflags`
  - `typedef` `traits_type::int_type` `int_type`
  - `typedef` `_ios_istate` `istate`
  - `typedef` `traits_type::off_type` `off_type`
  - `typedef` `_ios_Openmode` `openmode`
  - `typedef` `traits_type::pos_type` `pos_type`
  - `typedef` `_ios_Seekdir` `seekdir`
  - `typedef` `_Traits` `traits_type`
- 
- `typedef` `num_put<_CharT, ostreambuf_iterator<_CharT, _Traits>>` `__num_put_type`

#### Public Member Functions

- `basic_stringstream` ()
- `basic_stringstream` (`ios_base::openmode` `__m`)
- `basic_stringstream` (`const` `__string_type` `&__str`, `ios_base::openmode` `__m=ios_base::out|ios_base::in`)
- `basic_stringstream` (`const` `basic_stringstream` `&__rhs`)=delete
- `basic_stringstream` (`basic_stringstream` `&&__rhs`)
- `~basic_stringstream` ()
- `template<typename _ValueT>`  
`basic_istream<_CharT, _Traits>` `&_M_extract` (`_ValueT` `&__v`)
- `const` `locale` `&_M_getloc` () `const`
- `template<typename _ValueT>`  
`basic_ostream<_CharT, _Traits>` `&_M_insert` (`_ValueT` `__v`)
- `void` `_M_setstate` (`istate` `__state`)
- `bool` `bad` () `const`
- `void` `clear` (`istate` `__state=goodbit`)
- `basic_ios` `&copyfmt` (`const` `basic_ios` `&__rhs`)
- `bool` `eof` () `const`
- `istate` `exceptions` () `const`
- `void` `exceptions` (`istate` `__except`)
- `bool` `fail` () `const`
- `char_type` `fill` () `const`
- `char_type` `fill` (`char_type` `__ch`)
- `fmtflags` `flags` () `const`
- `fmtflags` `flags` (`fmtflags` `__fmtfl`)
- `__ostream_type` `&flush` ()
- `streamsize` `gcount` () `const`

- `template<>`  
`basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & getline (char_type *__s, streamsize __n, char_type __delim)`
  - `locale getloc () const`
  - `bool good () const`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< char > & ignore (streamsize __n, int_type __delim)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n)`
  - `template<>`  
`basic_istream< wchar_t > & ignore (streamsize __n, int_type __delim)`
  - `locale imbue (const locale &__loc)`
  - `long & iword (int __ix)`
  - `char narrow (char_type __c, char __dfault) const`
  - `__ostream_type & operator<< (const void *__p)`
  - `__ostream_type & operator<< (__streambuf_type *__sb)`
  - `basic_stringstream & operator= (const basic_stringstream &)=delete`
  - `basic_stringstream & operator= (basic_stringstream &&__rhs)`
  - `__istream_type & operator>> (void *&__p)`
  - `__istream_type & operator>> (__streambuf_type *__sb)`
  - `streamsize precision () const`
  - `streamsize precision (streamsize __prec)`
  - `void *& pword (int __ix)`
  - `basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)`
  - `__stringbuf_type * rdbuf () const`
  - `iosstate rdstate () const`
  - `void register_callback (event_callback __fn, int __index)`
  - `__ostream_type & seekp (pos_type)`
  - `__ostream_type & seekp (off_type, ios_base::seekdir)`
  - `fmtflags self (fmtflags __fmtfl)`
  - `fmtflags self (fmtflags __fmtfl, fmtflags __mask)`
  - `void setstate (iosstate __state)`
  - `__string_type str () const`
  - `void str (const __string_type &__s)`
  - `void swap (basic_stringstream &__rhs)`
  - `pos_type tellp ()`
  - `basic_ostream< _CharT, _Traits > * tie () const`
  - `basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)`
  - `void unsetf (fmtflags __mask)`
  - `char_type widen (char __c) const`
  - `streamsize width () const`
  - `streamsize width (streamsize __wide)`
- 
- `__istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))`
  - `__istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))`

- [\\_\\_istream\\_type](#) & [operator>>](#) ([ios\\_base](#) &(\*\_\_pf)([ios\\_base](#) &))

### Extractors

All the [operator>>](#) functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- [\\_\\_istream\\_type](#) & [operator>>](#) (`bool` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`short` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`unsigned short` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`int` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`unsigned int` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`long` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`unsigned long` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`long long` &\_\_n)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`unsigned long long` &\_\_n)
- 
- [\\_\\_istream\\_type](#) & [operator>>](#) (`float` &\_\_f)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`double` &\_\_f)
  - [\\_\\_istream\\_type](#) & [operator>>](#) (`long double` &\_\_f)

### Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_istream::sentry` with the second argument (`noskipws`) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by `gcount()`.

If an exception is thrown during extraction, `ios_base::badbit` will be turned on in the stream's error state (without causing an `ios_base::failure` to be thrown) and the original exception will be rethrown if `badbit` is set in the exceptions mask.

- `int_type` [get](#) ()
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` &\_\_c)
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [get](#) ([\\_\\_streambuf\\_type](#) & \_\_sb)
- [\\_\\_istream\\_type](#) & [getline](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n, `char_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [getline](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n, `int_type` \_\_delim)
- [\\_\\_istream\\_type](#) & [ignore](#) ([streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [ignore](#) ()
- `int_type` [peek](#) ()
- [\\_\\_istream\\_type](#) & [read](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [readsome](#) (`char_type` \* \_\_s, [streamsize](#) \_\_n)
- [\\_\\_istream\\_type](#) & [putback](#) (`char_type` \_\_c)

- `__istream_type & unget ()`
  - `int sync ()`
  - `pos_type tellg ()`
  - `__istream_type & seekg (pos_type)`
  - `__istream_type & seekg (off_type, ios_base::seekdir)`
- 
- `operator bool () const`
  - `bool operator! () const`
- 
- `__ostream_type & operator<< (__ostream_type &(__pf)(__ostream_type &))`
  - `__ostream_type & operator<< (__ios_type &(__pf)(__ios_type &))`
  - `__ostream_type & operator<< (ios_base &(__pf)(ios_base &))`

### Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state without causing an `ios_base::failure` to be thrown. The original exception will then be rethrown.

- `__ostream_type & operator<< (long __n)`
  - `__ostream_type & operator<< (unsigned long __n)`
  - `__ostream_type & operator<< (bool __n)`
  - `__ostream_type & operator<< (short __n)`
  - `__ostream_type & operator<< (unsigned short __n)`
  - `__ostream_type & operator<< (int __n)`
  - `__ostream_type & operator<< (unsigned int __n)`
  - `__ostream_type & operator<< (long long __n)`
  - `__ostream_type & operator<< (unsigned long long __n)`
- 
- `__ostream_type & operator<< (double __f)`
  - `__ostream_type & operator<< (float __f)`
  - `__ostream_type & operator<< (long double __f)`

### Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type `std::basic_ostream::sentry`. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, `ios_base::badbit` will be turned on in the stream's error state. If `badbit` is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- `__ostream_type & put (char_type __c)`
- `void _M_write (const char_type * __s, streamsize __n)`
- `__ostream_type & write (const char_type * __s, streamsize __n)`



### Static Public Member Functions

- static bool `sync_with_stdio` (bool \_\_sync=true)
- static int `xalloc` () throw ()

### Static Public Attributes

- static const `fmtflags` `adjustfield`
- static const `openmode` `app`
- static const `openmode` `ate`
- static const `iosstate` `badbit`
- static const `fmtflags` `basefield`
- static const `seekdir` `beg`
- static const `openmode` `binary`
- static const `fmtflags` `boolalpha`
- static const `seekdir` `cur`
- static const `fmtflags` `dec`
- static const `seekdir` `end`
- static const `iosstate` `eofbit`
- static const `iosstate` `failbit`
- static const `fmtflags` `fixed`
- static const `fmtflags` `floatfield`
- static const `iosstate` `goodbit`
- static const `fmtflags` `hex`
- static const `openmode` `in`
- static const `fmtflags` `internal`
- static const `fmtflags` `left`
- static const `fmtflags` `oct`
- static const `openmode` `out`
- static const `fmtflags` `right`
- static const `fmtflags` `scientific`
- static const `fmtflags` `showbase`
- static const `fmtflags` `showpoint`
- static const `fmtflags` `showpos`
- static const `fmtflags` `skipws`
- static const `openmode` `trunc`
- static const `fmtflags` `unitbuf`
- static const `fmtflags` `uppercase`

### Protected Types

- enum { `_S_local_word_size` }

## Protected Member Functions

- void **\_M\_cache\_locale** (const [locale](#) &\_\_loc)
- void **\_M\_call\_callbacks** ([event](#) \_\_ev) throw ()
- void **\_M\_dispose\_callbacks** (void) throw ()
- template<typename \_ValueT >  
  [\\_\\_istream\\_type](#) & **\_M\_extract** (\_ValueT &\_\_v)
- [\\_Words](#) & **\_M\_grow\_words** (int \_\_index, bool \_\_iword)
- void **\_M\_init** () throw ()
- template<typename \_ValueT >  
  [\\_\\_ostream\\_type](#) & **\_M\_insert** (\_ValueT \_\_v)
- void **\_M\_move** ([ios\\_base](#) &) noexcept
- void **\_M\_swap** ([ios\\_base](#) &\_\_rhs) noexcept
- void **init** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **move** ([basic\\_ios](#) &\_\_rhs)
- void **move** ([basic\\_ios](#) &&\_\_rhs)
- void **set\_rdbuf** ([basic\\_streambuf](#)< \_CharT, \_Traits > \*\_\_sb)
- void **swap** ([basic\\_ostream](#) &\_\_rhs)
- void **swap** ([basic\\_ios](#) &\_\_rhs) noexcept
- void **swap** ([basic\\_istream](#) &\_\_rhs)
- void **swap** ([basic\\_iostream](#) &\_\_rhs)

## Protected Attributes

- [\\_Callback\\_list](#) \* **\_M\_callbacks**
- const [\\_\\_ctype\\_type](#) \* **\_M\_ctype**
- [iostate](#) **\_M\_exception**
- [char\\_type](#) **\_M\_fill**
- bool **\_M\_fill\_init**
- [fmtflags](#) **\_M\_flags**
- [streamsize](#) **\_M\_gcount**
- [locale](#) **\_M\_ios\_locale**
- [\\_Words](#) **\_M\_local\_word** [[\\_S\\_local\\_word\\_size](#)]
- const [\\_\\_num\\_get\\_type](#) \* **\_M\_num\_get**
- const [\\_\\_num\\_put\\_type](#) \* **\_M\_num\_put**
- [streamsize](#) **\_M\_precision**
- [basic\\_streambuf](#)< \_CharT, \_Traits > \* **\_M\_streambuf**
- [iostate](#) **\_M\_streambuf\_state**
- [basic\\_ostream](#)< \_CharT, \_Traits > \* **\_M\_tie**
- [streamsize](#) **\_M\_width**
- [\\_Words](#) \* **\_M\_word**
- int **\_M\_word\_size**
- [\\_Words](#) **\_M\_word\_zero**

## 4.310.1 Detailed Description

```
template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringstream< _CharT, _Traits, _Alloc >
```

Controlling input and output for std::string.

## Template Parameters

|                      |                                                                                 |
|----------------------|---------------------------------------------------------------------------------|
| <code>_CharT</code>  | Type of character stream.                                                       |
| <code>_Traits</code> | Traits for character type, defaults to <code>char_traits&lt;_CharT&gt;</code> . |
| <code>_Alloc</code>  | Allocator type, defaults to <code>allocator&lt;_CharT&gt;</code> .              |

This class supports reading from and writing to objects of type `std::basic_string`, using the inherited functions from `std::basic_istream`. To control the associated sequence, an instance of `std::basic_stringbuf` is used, which this page refers to as `sb`.

Definition at line 108 of file `iosfwd`.

## 4.310.2 Member Typedef Documentation

## 4.310.2.1 \_\_num\_put\_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios<_CharT, _Traits>::__num_put_type [inherited]
```

These are non-standard types.

Definition at line 89 of file `basic_ios.h`.

## 4.310.2.2 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i) [inherited]
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.310.2.3 `fmtflags`

```
typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

## 4.310.2.4 iostate

```
typedef _Ios_Iostate std::ios_base::iostate [inherited]
```

This is a bitmask type.

*\_Ios\_Iostate* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *iostate* are:

- badbit
- eofbit
- failbit
- goodbit

Definition at line 416 of file *ios\_base.h*.

## 4.310.2.5 openmode

```
typedef _Ios_Openmode std::ios_base::openmode [inherited]
```

This is a bitmask type.

*\_Ios\_Openmode* is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type *openmode* are:

- app
- ate
- binary
- in
- out
- trunc

Definition at line 447 of file *ios\_base.h*.

#### 4.310.2.6 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.310.3 Member Enumeration Documentation

#### 4.310.3.1 event

```
enum std::ios_base::event [inherited]
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

### 4.310.4 Constructor & Destructor Documentation

#### 4.310.4.1 basic\_stringstream() [1/3]

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream () [inline]
```

Default constructor starts with an empty string buffer.

Initializes `sb` using the mode `in|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with `NULL`, because the string class does its own memory management.

Definition at line 741 of file `sstream`.

#### 4.310.4.2 basic\_stringstream() [2/3]

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
 ios_base::openmode __m) [inline], [explicit]
```

Starts with an empty string buffer.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__m</code> | Whether the buffer can read, or write, or both. |
|------------------|-------------------------------------------------|

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 756 of file `sstream`.

## 4.310.4.3 basic\_stringstream() [3/3]

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
 const __string_type & __str,
 ios_base::openmode __m = ios_base::out | ios_base::in) [inline], [explicit]
```

Starts with an existing string buffer.

## Parameters

|                    |                                                 |
|--------------------|-------------------------------------------------|
| <code>__str</code> | A string to copy as a starting buffer.          |
| <code>__m</code>   | Whether the buffer can read, or write, or both. |

Initializes `sb` using `__str` and `__m`, and passes `&sb` to the base class initializer.

That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

Definition at line 772 of file `sstream`.

## 4.310.4.4 ~basic\_stringstream()

```
template<typename _CharT , typename _Traits , typename _Alloc >
std::basic_stringstream< _CharT, _Traits, _Alloc >::~~basic_stringstream () [inline]
```

The destructor does nothing.

The buffer is deallocated by the `stringbuf` object, not the formatting stream.

Definition at line 783 of file `sstream`.

## 4.310.5 Member Function Documentation

#### 4.310.5.1 `_M_getloc()`

```
const locale& std::ios_base::_M_getloc () const [inline], [inherited]
```

Locale access.

##### Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _Inlter>::do_get()`, `std::money_get<_CharT, _Inlter>::do_get()`, `std::num_get<_CharT, _Inlter>::do_get()`, `std::time_get<_CharT, _Inlter>::do_get_date()`, `std::time_get<_CharT, _Inlter>::do_get_monthname()`, `std::time_get<_CharT, _Inlter>::do_get_time()`, `std::time_get<_CharT, _Inlter>::do_get_weekday()`, `std::time_put<_CharT, _Outlter>::do_put()`, `std::num_put<_CharT, _Outlter>::do_put()`, `std::time_get<_CharT, _Inlter>::get()`, and `std::time_put<_CharT, _Outlter>::put()`.

#### 4.310.5.2 `_M_write()`

```
template<typename _CharT, typename _Traits>
void std::basic_ostream<_CharT, _Traits>::_M_write (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Core write functionality, without sentry.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

Definition at line 317 of file `ostream`.

#### 4.310.5.3 `bad()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios<_CharT, _Traits>::bad () const [inline], [inherited]
```

Fast error checking.



**Returns**

True if the badbit is set.

Note that other iostate flags may also be set.

Definition at line 211 of file basic\_ios.h.

**4.310.5.4 clear()**

```
template<typename _CharT , typename _Traits >
void std::basic_ios< _CharT, _Traits >::clear (
 iostate __state = goodbit) [inherited]
```

[Re]sets the error state.

**Parameters**

|                      |                               |
|----------------------|-------------------------------|
| <code>__state</code> | The new state flag(s) to set. |
|----------------------|-------------------------------|

See std::ios\_base::iostate for the possible bit values. Most users will not need to pass an argument.

Definition at line 41 of file basic\_ios.tcc.

Referenced by std::basic\_ios< char, \_Traits >::exceptions(), std::\_\_detail::operator>>(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_ios< char, \_Traits >::setstate(), and std::basic\_istream< char >::unget().

**4.310.5.5 copyfmt()**

```
template<typename _CharT , typename _Traits >
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
 const basic_ios< _CharT, _Traits > & __rhs) [inherited]
```

Copies fields of `__rhs` into this.

**Parameters**

|                    |                                   |
|--------------------|-----------------------------------|
| <code>__rhs</code> | The source values for the copies. |
|--------------------|-----------------------------------|

**Returns**

Reference to this object.

All fields of `__rhs` are copied into this object except that `rdbuf()` and `rdstate()` remain unchanged. All values in the pword

and `iword` arrays are copied. Before copying, each callback is invoked with `erase_event`. After copying, each (new) callback is invoked with `copyfmt_event`. The final step is to copy exceptions().

Definition at line 63 of file `basic_ios.tcc`.

#### 4.310.5.6 `eof()`

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const [inline], [inherited]
```

Fast error checking.

##### Returns

True if the eofbit is set.

Note that other `iostate` flags may also be set.

Definition at line 190 of file `basic_ios.h`.

#### 4.310.5.7 `exceptions()` [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const [inline], [inherited]
```

Throwing exceptions on errors.

##### Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of `exceptions(iostate)` for the meaning of the return value.

Definition at line 222 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`.

#### 4.310.5.8 `exceptions()` [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
 iostate __except) [inline], [inherited]
```

Throwing exceptions on errors.

## Parameters

|                       |                          |
|-----------------------|--------------------------|
| <code>__except</code> | The new exceptions mask. |
|-----------------------|--------------------------|

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type `std::ios_base::failure` is thrown.

If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>
int main()
{
 std::set_terminate (__gnu_cxx::__verbose_terminate_handler);
 std::ifstream f ("/etc/motd");
 std::cerr << "Setting badbit\n";
 f.setstate (std::ios_base::badbit);
 std::cerr << "Setting exception mask\n";
 f.exceptions (std::ios_base::badbit);
}
```

Definition at line 257 of file `basic_ios.h`.

## 4.310.5.9 fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const [inline], [inherited]
```

Fast error checking.

## Returns

True if either the badbit or the failbit is set.

Checking the badbit in `fail()` is historical practice. Note that other iostate flags may also be set.

Definition at line 201 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::operator bool()`, and `std::basic_ios< char, _Traits >::operator!()`.

## 4.310.5.10 fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const [inline], [inherited]
```

Retrieves the *empty* character.

## Returns

The current fill character.

It defaults to a space ( ' ' ) in the current locale.

Definition at line 370 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::basic_ios< char, _Traits >::fill()`, and `std::operator<<()`.

**4.310.5.11** `fill()` [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
 char_type __ch) [inline], [inherited]
```

Sets a new *empty* character.

**Parameters**

|                   |                    |
|-------------------|--------------------|
| <code>__ch</code> | The new character. |
|-------------------|--------------------|

**Returns**

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via `setw`), Q characters are actually used, and Q<P. It defaults to a space ( ' ') in the current locale.

Definition at line 390 of file `basic_ios.h`.

**4.310.5.12** `flags()` [1/2]

```
fmtflags std::ios_base::flags () const [inline], [inherited]
```

Access to format flags.

**Returns**

The format control flags for both input and output.

Definition at line 649 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _OutIter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

**4.310.5.13** `flags()` [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags all at once.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

**Returns**

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file `ios_base.h`.

**4.310.5.14 flush()**

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::flush () [inherited]
```

Synchronizing the stream buffer.

**Returns**

\*this

If `rdbuf()` is a null pointer, changes nothing.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit`.

Definition at line 211 of file `ostream.tcc`.

**4.310.5.15 gcount()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream<_CharT, _Traits>::gcount () const [inline], [inherited]
```

Character counting.

**Returns**

The number of characters extracted by the previous unformatted input function dispatched for this stream.

Definition at line 269 of file `istream`.

**4.310.5.16** `get()` [1/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 244 of file istream.tcc.

**4.310.5.17** `get()` [2/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type & __c) [inherited]
```

Simple extraction.

**Parameters**

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The character in which to store data. |
|------------------|---------------------------------------|

**Returns**

\*this

Tries to extract a character and store it in \_\_c. If none are available, sets failbit and returns traits::eof().

**Note**

This function is not overloaded on signed char and unsigned char.

Definition at line 280 of file istream.tcc.

**4.310.5.18** `get()` [3/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Simple multiple-character extraction.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__s</code>     | Pointer to an array.                                        |
| <code>__n</code>     | Maximum number of characters to store in <code>__s</code> . |
| <code>__delim</code> | A "stop" character.                                         |

## Returns

`*this`

Characters are extracted and stored into `__s` until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals `__delim`, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.

In any case, a null character is stored into the next location in the array.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 317 of file istream.tcc.

4.310.5.19 `get()` [4/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::get (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Simple multiple-character extraction.

## Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__s</code> | Pointer to an array.                                      |
| <code>__n</code> | Maximum number of characters to store in <code>s</code> . |

**Returns**

\*this

Returns `get(__s,__n,widen("\n"))`.

Definition at line 354 of file `istream`.

**4.310.5.20 `get()`** [5/6]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb,
 char_type __delim) [inherited]
```

Extraction into another streambuf.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__sb</code>    | A streambuf in which to store data. |
| <code>__delim</code> | A "stop" character.                 |

**Returns**

\*this

Characters are extracted and inserted into `__sb` until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals `__delim` (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, `failbit` is set in the stream's error state.

Definition at line 364 of file `istream.tcc`.

**4.310.5.21 `get()`** [6/6]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::get (
 __streambuf_type & __sb) [inline], [inherited]
```

Extraction into another streambuf.



## Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <code>__sb</code> | A streambuf in which to store data. |
|-------------------|-------------------------------------|

## Returns

\*this

Returns `get(__sb, widen("\n"))`.

Definition at line 387 of file istream.

4.310.5.22 `getline()` [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

String extraction.

## Parameters

|                      |                                               |
|----------------------|-----------------------------------------------|
| <code>__s</code>     | A character array in which to store the data. |
| <code>__n</code>     | Maximum number of characters to extract.      |
| <code>__delim</code> | A "stop" character.                           |

## Returns

\*this

Extracts and stores characters into `__s` until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the `__s` array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)

In any case, a null character is stored in the next location in the array.

Definition at line 408 of file istream.tcc.

**4.310.5.23** `getline()` [2/3]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::getline (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

String extraction.

**Parameters**

|                  |                                               |
|------------------|-----------------------------------------------|
| <code>__s</code> | A character array in which to store the data. |
| <code>__n</code> | Maximum number of characters to extract.      |

**Returns**

`*this`

Returns `getline(__s,__n,widen("\n"))`.

Definition at line 427 of file `istream`.

**4.310.5.24** `getline()` [3/3]

```
template<>
basic_istream< char > & std::basic_istream< char >::getline (
 char_type * __s,
 streamsize __n,
 char_type __delim) [inherited]
```

Explicit specialization declarations, defined in `src/istream.cc`.

**4.310.5.25** `getloc()`

```
locale std::ios_base::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::money_put< _CharT, _Outiter >::do_put()`, and `std::ws()`.

## 4.310.5.26 good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const [inline], [inherited]
```

Fast error checking.

## Returns

True if no error flags are set.

A wrapper around rdbuf.

Definition at line 180 of file basic\_ios.h.

Referenced by std::\_\_detail::operator>>(), std::basic\_ostream< \_CharT, \_Traits >::sentry::sentry(), and std::basic\_istream< \_CharT, \_Traits >::sentry::sentry().

## 4.310.5.27 ignore() [1/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n,
 int_type __delim) [inherited]
```

Discarding characters.

## Parameters

|                      |                                  |
|----------------------|----------------------------------|
| <code>__n</code>     | Number of characters to discard. |
| <code>__delim</code> | A "stop" character.              |

## Returns

\*this

Extracts characters and throws them away until one of the following happens:

- if `__n != std::numeric_limits<int>::max()`, `__n` characters are extracted
- the input sequence reaches end-of-file
- the next character equals `__delim` (in this case, the character is extracted); note that this condition will never occur if `__delim` equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

Definition at line 563 of file istream.tcc.

**4.310.5.28 ignore()** [2/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 streamsize __n) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 501 of file istream.tcc.

**4.310.5.29 ignore()** [3/3]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
 void) [inherited]
```

Simple extraction.

**Returns**

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

Definition at line 468 of file istream.tcc.

**4.310.5.30 imbue()**

```
template<typename _CharT , typename _Traits >
locale std::basic_ios< _CharT, _Traits >::imbue (
 const locale & __loc) [inherited]
```

Moves to a new locale.

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.

Additional l10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 114 of file `basic_ios.tcc`.

**4.310.5.31 init()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios<_CharT, _Traits>::init (
 basic_streambuf<_CharT, _Traits> * __sb) [protected], [inherited]
```

All setup is performed here.

This is called from the public constructor. It is not virtual and cannot be redefined.

Definition at line 126 of file `basic_ios.tcc`.

Referenced by `std::basic_ios<char, _Traits>::basic_ios()`.

**4.310.5.32 iword()**

```
long& std::ios_base::iword (
 int __ix) [inline], [inherited]
```

Access to integer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file ios\_base.h.

#### 4.310.5.33 narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Squeezes characters.

##### Parameters

|                        |                          |
|------------------------|--------------------------|
| <code>__c</code>       | The character to narrow. |
| <code>__default</code> | The character to narrow. |

##### Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.

Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,default)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>.

Definition at line 430 of file basic\_ios.h.

#### 4.310.5.34 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

Definition at line 117 of file basic\_ios.h.

**4.310.5.35 operator!()**

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [inherited]
```

The quick-and-easy status check.

This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`.

Definition at line 125 of file `basic_ios.h`.

**4.310.5.36 operator<<()** [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 __ostream_type &(*) (__ostream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 108 of file `ostream`.

**4.310.5.37 operator<<()** [2/17]

```
template<typename _CharT, typename _Traits>
__ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.

Definition at line 117 of file `ostream`.

**4.310.5.38 operator<<()** [3/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like `"std::cout << std::endl"`. For more information, see the `omanip` header.


Definition at line 127 of file `ostream`.

**4.310.5.39 operator<<()** [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                   |                                      |
|-----------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                  |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.


Definition at line 166 of file `ostream`.

**4.310.5.40 operator<<() [5/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                                                                     |                                      |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | A variable of builtin integral type. |
| <code>__n</code>                                                                    |                                      |

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 170 of file `ostream`.

**4.310.5.41 operator<<() [6/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 bool __n) [inline], [inherited]
```

Integer arithmetic inserters.



## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 174 of file ostream.

## 4.310.5.42 operator&lt;&lt;() [7/17]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::operator<< (
 short __n) [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 92 of file ostream.tcc.

## 4.310.5.43 operator&lt;&lt;() [8/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned short __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>__</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 181 of file `ostream`.

**4.310.5.44 `operator<<()` [9/17]**

```
template<typename _CharT, typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 int __n) [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                                     |                                      |
|-------------------------------------|--------------------------------------|
| <code>__</code><br><code>__n</code> | A variable of builtin integral type. |
|-------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 106 of file `ostream.tcc`.

**4.310.5.45 `operator<<()` [10/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 unsigned int __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 192 of file ostream.

## 4.310.5.46 operator&lt;&lt;() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

## Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__n</code> | A variable of builtin integral type. |
|------------------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 201 of file ostream.

## 4.310.5.47 operator&lt;&lt;() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream<_CharT, _Traits>::operator<< (
 unsigned long long __n) [inline], [inherited]
```

Integer arithmetic inserters.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 205 of file `ostream`.

**4.310.5.48 operator<<() [13/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 220 of file `ostream`.

**4.310.5.49 operator<<() [14/17]**

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 float __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 224 of file ostream.

## 4.310.5.50 operator&lt;&lt;() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 long double __f) [inline], [inherited]
```

Floating point arithmetic inserters.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 232 of file ostream.

## 4.310.5.51 operator&lt;&lt;() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type& std::basic_ostream< _CharT, _Traits >::operator<< (
 const void * __p) [inline], [inherited]
```

Pointer arithmetic inserters.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

Definition at line 245 of file `ostream`.

**4.310.5.52 operator<<() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
 __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

Definition at line 120 of file `ostream.tcc`.

## 4.310.5.53 operator&gt;&gt;() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __istream_type &(*) (__istream_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 120 of file `istream`.

## 4.310.5.54 operator&gt;&gt;() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 __ios_type &(*) (__ios_type &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.

Definition at line 124 of file `istream`.

## 4.310.5.55 operator&gt;&gt;() [3/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 ios_base &(*) (ios_base &) __pf) [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the `io manip` header.


Definition at line 131 of file `istream`.

## 4.310.5.56 operator&gt;&gt;() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                   |                                      |
|---------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>_n</code> | A variable of builtin integral type. |
|---------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.


Definition at line 168 of file `istream`.

**4.310.5.57 operator>>() [5/17]**

```
template<typename _CharT, typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 short & __n) [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                                                                                     |                                      |
|-----------------------------------------------------------------------------------------------------|--------------------------------------|
|  <code>_n</code> | A variable of builtin integral type. |
|-----------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 122 of file `istream.tcc`.

**4.310.5.58 operator>>() [6/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned short & __n) [inline], [inherited]
```

Integer arithmetic extractors.



## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 175 of file istream.

## 4.310.5.59 operator&gt;&gt;() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::operator>> (
 int & __n) [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 167 of file istream.tcc.

## 4.310.5.60 operator&gt;&gt;() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned int & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                  |                                      |
|----------------------------------|--------------------------------------|
| $\leftarrow$<br><code>__n</code> | A variable of builtin integral type. |
|----------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 182 of file `istream`.

**4.310.5.61 `operator>>()` [9/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                                  |                                      |
|----------------------------------|--------------------------------------|
| $\leftarrow$<br><code>__n</code> | A variable of builtin integral type. |
|----------------------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 186 of file `istream`.

**4.310.5.62 `operator>>()` [10/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 unsigned long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 190 of file `istream`.

## 4.310.5.63 operator&gt;&gt;() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

## Parameters

|       |                                      |
|-------|--------------------------------------|
| $\_n$ | A variable of builtin integral type. |
|-------|--------------------------------------|

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 195 of file `istream`.

## 4.310.5.64 operator&gt;&gt;() [12/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream<_CharT, _Traits>::operator>> (
 unsigned long long & __n) [inline], [inherited]
```

Integer arithmetic extractors.

**Parameters**

|                       |                                      |
|-----------------------|--------------------------------------|
| $\leftarrow$<br>$\_n$ | A variable of builtin integral type. |
|-----------------------|--------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 199 of file `istream`.

**4.310.5.65 `operator>>()` [13/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 float & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

**Parameters**

|                                                     |                                            |
|-----------------------------------------------------|--------------------------------------------|
| $\leftarrow$<br>$\_$<br>$\leftarrow$<br>$\_$<br>$f$ | A variable of builtin floating point type. |
|-----------------------------------------------------|--------------------------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 214 of file `istream`.

**4.310.5.66 `operator>>()` [14/17]**

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 218 of file `istream`.

## 4.310.5.67 operator&gt;&gt;() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 long double & __f) [inline], [inherited]
```

Floating point arithmetic extractors.

## Parameters

|          |                                            |
|----------|--------------------------------------------|
| ↩        | A variable of builtin floating point type. |
| ↩        |                                            |
| ↩        |                                            |
| ↩        |                                            |
| <i>f</i> |                                            |

## Returns

\*this if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 222 of file `istream`.

## 4.310.5.68 operator&gt;&gt;() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type& std::basic_istream< _CharT, _Traits >::operator>> (
 void *& __p) [inline], [inherited]
```

Basic arithmetic extractors.

**Parameters**

|                        |                             |
|------------------------|-----------------------------|
| <code>_↵<br/>_p</code> | A variable of pointer type. |
|------------------------|-----------------------------|

**Returns**

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

Definition at line 235 of file `istream`.

**4.310.5.69 operator>>() [17/17]**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
 __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

**Parameters**

|                   |                          |
|-------------------|--------------------------|
| <code>__sb</code> | A pointer to a streambuf |
|-------------------|--------------------------|

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If `__sb` is `NULL`, the stream will set failbit in its error state.

Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

Definition at line 212 of file `istream.tcc`.

## 4.310.5.70 peek()

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::int_type std::basic_istream<_CharT, _Traits>::peek (
 void) [inherited]
```

Looking ahead in the stream.

## Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

Definition at line 628 of file `istream.tcc`.

## 4.310.5.71 precision() [1/2]

```
streamsize std::ios_base::precision () const [inline], [inherited]
```

Flags access.

## Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, and `std::operator<<()`.

## 4.310.5.72 precision() [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

**Returns**

The previous value of `precision()`.


Definition at line 728 of file `ios_base.h`.

**4.310.5.73 put()**

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
 char_type __c) [inherited]
```

Simple insertion.

**Parameters**

|                                                                                   |                          |
|-----------------------------------------------------------------------------------|--------------------------|
|  | The character to insert. |
| <code>__c</code>                                                                  |                          |

**Returns**

`*this`

Tries to insert `__c`.

**Note**

This function is not overloaded on signed char and unsigned char.


Definition at line 149 of file `ostream.tcc`.

**4.310.5.74 putback()**

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
 char_type __c) [inherited]
```

Unextracting a single character.

**Parameters**

|                                                                                     |                                                   |
|-------------------------------------------------------------------------------------|---------------------------------------------------|
|  | The character to push back into the input stream. |
| <code>__c</code>                                                                    |                                                   |



**Returns**

\*this

If `rdbuf()` is not null, calls `rdbuf() -> sputbackc(c)`.

If `rdbuf()` is null or if `sputbackc()` fails, sets `badbit` in the error state.

**Note**

This function first clears `eofbit`. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

Definition at line 719 of file `istream.tcc`.

**4.310.5.75 pword()**

```
void*& std::ios_base::pword (
 int __ix) [inline], [inherited]
```

Access to void pointer array.

**Parameters**

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

**Returns**

A reference to a `void*` associated with the index.

The `pword` function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file `ios_base.h`.

**4.310.5.76 rdbuf()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf<_CharT, _Traits> * std::basic_ios<_CharT, _Traits>::rdbuf (
 basic_streambuf<_CharT, _Traits> * __sb) [inherited]
```

Changing the underlying buffer.

**Parameters**

|                   |                        |
|-------------------|------------------------|
| <code>__sb</code> | The new stream buffer. |
|-------------------|------------------------|

**Returns**

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.

Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream foo; // or some other derived type
std::streambuf* p =;
foo.ios::rdbuf(p); // ios == basic_ios<char>
```

Definition at line 53 of file `basic_ios.tcc`.

**4.310.5.77 `rdbuf()`** [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc >
__stringbuf_type* std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf () const [inline]
```

Accessing the underlying buffer.

**Returns**

The current `basic_stringbuf` buffer.

This hides both signatures of `std::basic_ios::rdbuf()`.

Definition at line 823 of file `sstream`.

**4.310.5.78 `rdstate()`**

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const [inline], [inherited]
```

Returns the error state of the stream buffer.

**Returns**

A bit pattern (well, isn't everything?)

See `std::ios_base::iostate` for the possible bit values. Most users will call one of the interpreting wrappers, e.g., `good()`.

Definition at line 137 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::bad()`, `std::basic_ios< char, _Traits >::eof()`, `std::basic_ios< char, _Traits >::fail()`, `std::basic_ios< char, _Traits >::good()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::seekg()`, `std::basic_ios< char, _Traits >::setstate()`, and `std::basic_istream< char >::unget()`.

## 4.310.5.79 read()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction without delimiters.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

## Returns

\*this

If the stream state is `good()`, extracts characters and stores them into `__s` until one of the following happens:

- `__n` characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 658 of file `istream.tcc`.

## 4.310.5.80 readsome()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_istream< _CharT, _Traits >::readsome (
 char_type * __s,
 streamsize __n) [inherited]
```

Extraction until the buffer is exhausted, but no more.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A character array.                     |
| <code>__n</code> | Maximum number of characters to store. |

**Returns**

The number of characters extracted.

Extracts characters and stores them into `__s` depending on the number of characters remaining in the streambuf's buffer, `rdbuf() -> in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A, n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

Definition at line 687 of file `istream.tcc`.

**4.310.5.81 register\_callback()**

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index) [inherited]
```

Add the callback `__fn` with parameter `__index`.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.310.5.82 seekg()** [1/2]

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
 pos_type __pos) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 853 of file `istream.tcc`.

**4.310.5.83 seekg()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits> & std::basic_istream<_CharT, _Traits>::seekg (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current read position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

\*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off, __dir)`. If that function fails, sets failbit.

**Note**

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 892 of file `istream.tcc`.

**4.310.5.84 seekp()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits> & std::basic_ostream<_CharT, _Traits>::seekp (
 pos_type __pos) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__pos</code> | A file position object. |
|--------------------|-------------------------|

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

Definition at line 258 of file `ostream.tcc`.

**4.310.5.85 seekp()** [2/2]

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
 off_type __off,
 ios_base::seekdir __dir) [inherited]
```

Changing the current write position.

**Parameters**

|                    |                                 |
|--------------------|---------------------------------|
| <code>__off</code> | A file offset object.           |
| <code>__dir</code> | The direction in which to seek. |

**Returns**

`*this`

If `fail()` is not true, calls `rdbuf()->pubseekoff(off, dir)`. If that function fails, sets failbit.

Definition at line 290 of file `ostream.tcc`.

**4.310.5.86 setf()** [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file ios\_base.h.

Referenced by std::\_\_detail::operator>>().

**4.310.5.87 setf()** [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline], [inherited]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file ios\_base.h.

**4.310.5.88 setstate()**

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
 iostate __state) [inline], [inherited]
```

Sets additional flags in the error state.

**Parameters**

|                      |                                      |
|----------------------|--------------------------------------|
| <code>__state</code> | The additional state flag(s) to set. |
|----------------------|--------------------------------------|

See `std::ios_base::iostate` for the possible bit values.

Definition at line 157 of file `basic_ios.h`.

Referenced by `std::operator<<()`, `std::basic_ostream<_CharT, _Traits>::sentry::sentry()`, and `std::ws()`.

#### 4.310.5.89 `str()` [1/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
__string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str () const [inline]
```

Copying out the string buffer.

##### Returns

`rdbuf() -> str()`

Definition at line 831 of file `sstream`.

#### 4.310.5.90 `str()` [2/2]

```
template<typename _CharT , typename _Traits , typename _Alloc >
void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
 const __string_type & __s) [inline]
```

Setting a new buffer.

##### Parameters

|                  |                                      |
|------------------|--------------------------------------|
| <code>__s</code> | The string to use as a new sequence. |
|------------------|--------------------------------------|

Calls `rdbuf() -> str(s)`.

Definition at line 841 of file `sstream`.

#### 4.310.5.91 `sync()`

```
template<typename _CharT , typename _Traits >
int std::basic_istream< _CharT, _Traits >::sync (
 void) [inherited]
```

Synchronizing the stream buffer.



**Returns**

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.

Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets `badbit` and returns -1.

Otherwise, returns 0.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

Definition at line 789 of file `istream.tcc`.

**4.310.5.92 sync\_with\_stdio()**

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static], [inherited]
```

Interaction with the standard C I/O objects.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

**Returns**

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

**4.310.5.93 tellg()**

```
template<typename _CharT, typename _Traits>
basic_istream<_CharT, _Traits>::pos_type std::basic_istream<_CharT, _Traits>::tellg (
 void) [inherited]
```

Getting the current read position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, in)`.

**Note**

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with `putback`, `unget` and `seekg`, `eofbit` is not cleared first.

Definition at line 825 of file `istream.tcc`.

**4.310.5.94 tellp()**

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
[inherited]
```

Getting the current write position.

**Returns**

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0, cur, out)`.

Definition at line 237 of file `ostream.tcc`.

**4.310.5.95 tie()** [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie () const [inline], [inherited]
```

Fetches the current *tied* stream.

**Returns**

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

Definition at line 295 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::basic_ostream< _CharT, _Traits >::sentry::sentry()`.

**4.310.5.96 tie()** [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream<_CharT, _Traits>* std::basic_ios< _CharT, _Traits >::tie (
 basic_ostream< _CharT, _Traits > * __tiestr) [inline], [inherited]
```

Ties this stream to an output stream.

## Parameters

|                       |                    |
|-----------------------|--------------------|
| <code>__tiestr</code> | The output stream. |
|-----------------------|--------------------|

## Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

Definition at line 307 of file basic\_ios.h.

## 4.310.5.97 unget()

```
template<typename _CharT , typename _Traits >
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
 void) [inherited]
```

Unextracting the previous character.

## Returns

\*this

If rdbuf() is not null, calls rdbuf()->sungetc(c).

If rdbuf() is null or if sungetc() fails, sets badbit in the error state.

## Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

Definition at line 754 of file istream.tcc.

Referenced by std::\_\_detail::operator>>().

## 4.310.5.98 unsetf()

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline], [inherited]
```

Clearing format flags.

**Parameters**

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

**4.310.5.99 widen()**

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
 char __c) const [inline], [inherited]
```

Widens characters.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The character to widen. |
|------------------|-------------------------|

**Returns**

The widened character.

Maps a character of `char` to a character of `char_type`.

Returns the result of

```
std::use_facet<ctype<char_type>> >(getloc()).widen(c)
```

Additional I10n notes are at <http://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html>

Definition at line 449 of file `basic_ios.h`.

Referenced by `std::basic_ios< char, _Traits >::fill()`, `std::getline()`, `std::operator<<()`, and `std::tr2::operator>>()`.

**4.310.5.100 width()** [1/2]

```
streamsize std::ios_base::width () const [inline], [inherited]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIt >::do_put()`.

## 4.310.5.101 width() [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline], [inherited]
```

Changing flags.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

## Returns

The previous value of width().

Definition at line 751 of file ios\_base.h.

## 4.310.5.102 write()

```
template<typename _CharT , typename _Traits >
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
 const char_type * __s,
 streamsize __n) [inherited]
```

Character string insertion.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | The array to insert.                    |
| <code>__n</code> | Maximum number of characters to insert. |

## Returns

\*this

Characters are copied from `__s` and inserted into the stream until one of the following happens:

- `__n` characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

## Note

This function is not overloaded on signed char and unsigned char.

Definition at line 183 of file ostream.tcc.

#### 4.310.5.103 xalloc()

```
static int std::ios_base::xalloc () throw () [static], [inherited]
```

Access to unique indices.

##### Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.

#### 4.310.6 Member Data Documentation

##### 4.310.6.1 \_M\_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see `gcount()`.

Definition at line 82 of file `istream`.

Referenced by `std::basic_istream< char >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic_istream< char >::peek()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsomewhat()`, and `std::basic_istream< char >::unget()`.

##### 4.310.6.2 adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of `left|right|internal`. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _Outiter >::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

#### 4.310.6.3 app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.

Definition at line 450 of file ios\_base.h.

#### 4.310.6.4 ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.

Definition at line 453 of file ios\_base.h.

#### 4.310.6.5 badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file ios\_base.h.

Referenced by std::basic\_ios< char, \_Traits >::bad(), std::basic\_ios< char, \_Traits >::fail(), and std::operator<<().

#### 4.310.6.6 basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of setf.

Definition at line 399 of file ios\_base.h.

Referenced by std::dec(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), std::oct(), and std::basic\_ostream< char >::operator<<().

#### 4.310.6.7 beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file ios\_base.h.

#### 4.310.6.8 binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>.

Definition at line 458 of file ios\_base.h.

#### 4.310.6.9 boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file ios\_base.h.

Referenced by `std::boolalpha()`, `std::num_get<_CharT, _InIter >::do_get()`, `std::num_put<_CharT, _OutIter >::do_put()`, and `std::noboolalpha()`.

#### 4.310.6.10 cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc >::seekoff()`.

#### 4.310.6.11 dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by `std::dec()`.



## 4.310.6.12 end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>::seekoff().

## 4.310.6.13 eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get<\_CharT, \_InIter>::do\_get(), std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_date(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_time(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::basic\_ios<char, \_Traits>::eof(), std::time\_get<\_CharT, \_InIter>::get(), std::basic\_istream<char>::putback(), std::basic\_istream<char>::seekg(), std::basic\_istream<char>::unget(), and std::ws().

## 4.310.6.14 failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file ios\_base.h.

Referenced by std::num\_get<\_CharT, \_InIter>::do\_get(), std::time\_get<\_CharT, \_InIter>::do\_get\_monthname(), std::time\_get<\_CharT, \_InIter>::do\_get\_weekday(), std::time\_get<\_CharT, \_InIter>::do\_get\_year(), std::basic\_ios<char, \_Traits>::fail(), std::time\_get<\_CharT, \_InIter>::get(), and std::basic\_ostream<\_CharT, \_Traits>::sentry() and std::basic\_ostream<\_CharT, \_Traits>::flush().

## 4.310.6.15 fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file ios\_base.h.

Referenced by std::fixed(), and std::hexfloat().

#### 4.310.6.16 floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.310.6.17 goodbit

```
const iostate std::ios_base::goodbit [static], [inherited]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::num_get<_CharT, _InIter>::do_get()`, `std::time_get<_CharT, _InIter>::do_get_monthname()`, `std::time_get<_CharT, _InIter>::do_get_weekday()`, `std::time_get<_CharT, _InIter>::do_get_year()`, `std::basic_ostream<char>::flush()`, `std::basic_istream<char>::get()`, `std::time_get<_CharT, _InIter>::get()`, `std::basic_istream<char>::getline()`, `std::basic_istream<char>::ignore()`, `std::basic_ostream<char>::operator<<()`, `std::basic_istream<char>::operator>>()`, `std::operator>>()`, `std::basic_istream<char>::peek()`, `std::basic_ostream<char>::put()`, `std::basic_istream<char>::putback()`, `std::basic_istream<char>::read()`, `std::basic_istream<char>::readsomewhat()`, `std::basic_istream<char>::seekg()`, `std::basic_ostream<char>::seekp()`, `std::basic_istream<_CharT, _Traits>::sentry::sentry()`, `std::basic_istream<char>::sync()`, and `std::basic_istream<char>::unget()`.

#### 4.310.6.18 hex

```
const fmtflags std::ios_base::hex [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file `ios_base.h`.

Referenced by `std::num_get<_CharT, _InIter>::do_get()`, `std::num_put<_CharT, _OutIter>::do_put()`, `std::hex()`, and `std::basic_ostream<char>::operator<<()`.

#### 4.310.6.19 in

```
const openmode std::ios_base::in [static], [inherited]
```

Open for input. Default for `ifstream` and `fstream`.

Definition at line 461 of file `ios_base.h`.

Referenced by `std::basic_stringbuf<_CharT, _Traits, _Alloc>::overflow()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekoff()`, `std::basic_stringbuf<_CharT, _Traits, _Alloc>::seekpos()`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>::underflow()`.

#### 4.310.6.20 internal

```
const fmtflags std::ios_base::internal [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Definition at line 358 of file `ios_base.h`.

Referenced by `std::internal()`.

#### 4.310.6.21 left

```
const fmtflags std::ios_base::left [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file `ios_base.h`.

Referenced by `std::num_put< _CharT, _OutIter >::do_put()`, and `std::left()`.

#### 4.310.6.22 oct

```
const fmtflags std::ios_base::oct [static], [inherited]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.310.6.23 out

```
const openmode std::ios_base::out [static], [inherited]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

#### 4.310.6.24 right

```
const fmtflags std::ios_base::right [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file ios\_base.h.

Referenced by std::right().

#### 4.310.6.25 scientific

```
const fmtflags std::ios_base::scientific [static], [inherited]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file ios\_base.h.

Referenced by std::hexfloat(), and std::scientific().

#### 4.310.6.26 showbase

```
const fmtflags std::ios_base::showbase [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put<\_CharT, \_OutIter >::do\_put(), std::noshowbase(), and std::showbase().

#### 4.310.6.27 showpoint

```
const fmtflags std::ios_base::showpoint [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.310.6.28 showpos

```
const fmtflags std::ios_base::showpos [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.310.6.29 skipws

```
const fmtflags std::ios_base::skipws [static], [inherited]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.310.6.30 trunc

```
const openmode std::ios_base::trunc [static], [inherited]
```

Truncate an existing stream when opening. Default for ofstream.

Definition at line 467 of file ios\_base.h.

#### 4.310.6.31 unitbuf

```
const fmtflags std::ios_base::unitbuf [static], [inherited]
```

Flushes output after each output operation.

Definition at line 389 of file ios\_base.h.

Referenced by std::nunitbuf(), and std::unitbuf().

#### 4.310.6.32 uppercase

```
const fmtflags std::ios_base::uppercase [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _Outlter>::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [sstream](#)

### 4.311 std::bernoulli\_distribution Class Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef bool [result\\_type](#)

#### Public Member Functions

- [bernoulli\\_distribution](#) ()
- [bernoulli\\_distribution](#) (double \_\_p)
- **[bernoulli\\_distribution](#)** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double [p](#) () const
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool `operator==` (const `bernoulli_distribution` &\_\_d1, const `bernoulli_distribution` &\_\_d2)

## 4.311.1 Detailed Description

A Bernoulli random number distribution.

Generates a sequence of true and false values with likelihood  $p$  that true will come up and  $(1 - p)$  that false will appear.

Definition at line 3521 of file random.h.

## 4.311.2 Member Typedef Documentation

## 4.311.2.1 result\_type

```
typedef bool std::bernoulli_distribution::result_type
```

The type of the range of the distribution.

Definition at line 3525 of file random.h.

## 4.311.3 Constructor &amp; Destructor Documentation

## 4.311.3.1 bernoulli\_distribution() [1/2]

```
std::bernoulli_distribution::bernoulli_distribution () [inline]
```

Constructs a Bernoulli distribution with likelihood 0.5.

Definition at line 3561 of file random.h.

## 4.311.3.2 bernoulli\_distribution() [2/2]

```
std::bernoulli_distribution::bernoulli_distribution (
 double __p) [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood  $p$ .

**Parameters**

|                     |                                                                                         |
|---------------------|-----------------------------------------------------------------------------------------|
| $\leftarrow$<br>$p$ | [IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$ . |
|---------------------|-----------------------------------------------------------------------------------------|

Definition at line 3570 of file random.h.

**4.311.4 Member Function Documentation****4.311.4.1 max()**

```
result_type std::bernoulli_distribution::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3620 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

**4.311.4.2 min()**

```
result_type std::bernoulli_distribution::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3613 of file random.h.

References `std::numeric_limits<_Tp>::min()`.

**4.311.4.3 operator()()**

```
template<typename _UniformRandomNumberGenerator >
result_type std::bernoulli_distribution::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 3628 of file random.h.



#### 4.311.4.4 p()

```
double std::bernoulli_distribution::p () const [inline]
```

Returns the `p` parameter of the distribution.

Definition at line 3591 of file random.h.

#### 4.311.4.5 param() [1/2]

```
param_type std::bernoulli_distribution::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3598 of file random.h.

Referenced by `std::operator>>()`.

#### 4.311.4.6 param() [2/2]

```
void std::bernoulli_distribution::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3606 of file random.h.

#### 4.311.4.7 reset()

```
void std::bernoulli_distribution::reset () [inline]
```

Resets the distribution state.

Does nothing for a Bernoulli distribution.

Definition at line 3585 of file random.h.

#### 4.311.5 Friends And Related Function Documentation

#### 4.311.5.1 operator==

```
bool operator== (
 const bernoulli_distribution & __d1,
 const bernoulli_distribution & __d2) [friend]
```

Return true if two Bernoulli distributions have the same parameters.

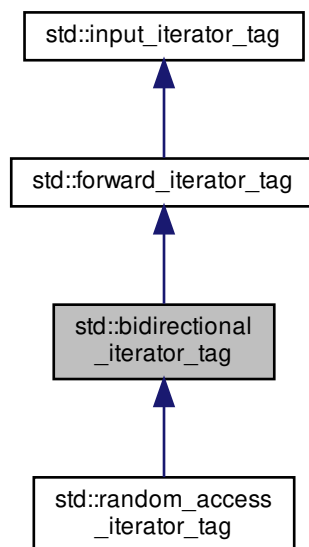
Definition at line 3670 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.312 std::bidirectional\_iterator\_tag Struct Reference

Inheritance diagram for std::bidirectional\_iterator\_tag:



##### 4.312.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.

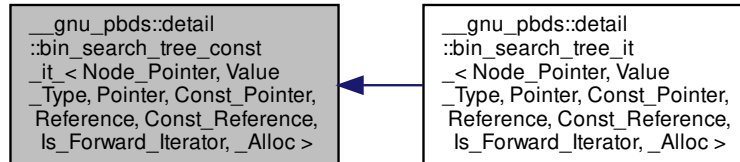
Definition at line 103 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.313 `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



#### Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::bidirectional_iterator_tag` **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

#### Public Member Functions

- **bin\_search\_tree\_const\_it\_** (const Node\_Pointer p\_nd=0)
- **bin\_search\_tree\_const\_it\_** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const
- const\_reference **operator\*** () const
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator++** ()
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) **operator++** (int)
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator--** ()
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) **operator--** (int)
- const\_pointer **operator->** () const

- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other)
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, !Is\\_Forward\\_Iterator, \\_Alloc >](#) &other)
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, !Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const

#### Public Attributes

- Node\_Pointer **m\_p\_nd**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 4.313.1 Detailed Description

template<typename Node\_Pointer, typename Value\_Type, typename Pointer, typename Const\_Pointer, typename Reference, typename Const\_Reference, bool Is\_Forward\_Iterator, typename \_Alloc>

class \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_it\_< Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc >

Const iterator.

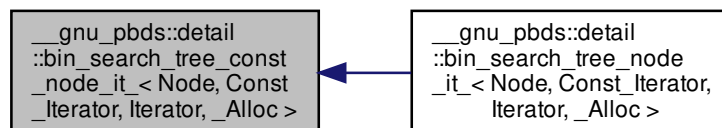
Definition at line 105 of file point\_iterators.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 4.314 \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >:



## Public Types

- typedef Const\_Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits](#)< \_Alloc, [metadata\\_type](#) >::[const\\_reference](#) [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Const\_Iterator [reference](#)
- typedef Const\_Iterator [value\\_type](#)

## Public Member Functions

- [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > &other) const
- [const\\_reference](#) [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_](#) < Node, Const\_Iterator, Iterator, \_Alloc > &other) const

## Public Attributes

- node\_pointer [m\\_p\\_nd](#)

### 4.314.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Const node iterator.

Definition at line 58 of file `bin_search_tree_/node_iterators.hpp`.

### 4.314.2 Member Typedef Documentation

#### 4.314.2.1 `const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::const_reference
```

Iterator's `__const` reference type.

Definition at line 77 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.314.2.2 difference\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type
```

Difference type.

Definition at line 68 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.2.3 iterator\_category

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::iterator_category
```

Category.

Definition at line 65 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.2.4 metadata\_const\_reference

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef rebind_traits<_Alloc, metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

Definition at line 84 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.2.5 metadata\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

Definition at line 80 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.2.6 reference

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::reference
```

Iterator's reference type.

Definition at line 74 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.2.7 value\_type

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
```

Iterator's value type.

Definition at line 71 of file bin\_search\_tree\_/node\_iterators.hpp.

### 4.314.3 Member Function Documentation

#### 4.314.3.1 get\_l\_child()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
```

Returns the \_\_const node iterator associated with the left node.

Definition at line 103 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.3.2 get\_metadata()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline]
```

Metadata access.

Definition at line 98 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.314.3.3 `get_r_child()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_const_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_const_
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
```

Returns the `__const` node iterator associated with the right node.

Definition at line 108 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.314.3.4 `operator!=(())`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!=((
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline]
```

Compares (negatively) to a different iterator object.

Definition at line 118 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.314.3.5 `operator*()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* () const [inline]
```

Access.

Definition at line 93 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.314.3.6 `operator==(())`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator==((
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline]
```

Compares to a different iterator object.

Definition at line 113 of file `bin_search_tree_/node_iterators.hpp`.

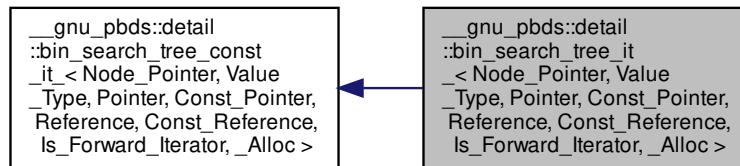
The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)



4.315 `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >`:



## Public Types

- typedef Const\_Pointer **const\_pointer**
- typedef Const\_Reference **const\_reference**
- typedef \_Alloc::difference\_type **difference\_type**
- typedef [std::bidirectional\\_iterator\\_tag](#) **iterator\_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value\_Type **value\_type**

## Public Member Functions

- **bin\_search\_tree\_it\_** (const Node\_Pointer p\_nd=0)
- **bin\_search\_tree\_it\_** (const [bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other)
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const
- bool **operator!=** (const [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) &other) const
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)::reference **operator\*** () const
- [bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator++** ()
- [bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator++** (int)
- [bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator--** ()
- [bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#) & **operator--** (int)
- [bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#)::pointer **operator->** () const

- [bin\\_search\\_tree\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↵ Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_↵ \_Pointer, Reference, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other)
- [bin\\_search\\_tree\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_Reference, Is\_↵ Forward\_Iterator, \_Alloc > & **operator=** (const [bin\\_search\\_tree\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_↵ \_Pointer, Reference, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other)
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Refer-  
ence, Const\_Reference, Is\_Forward\_Iterator, \_Alloc > &other) const
- bool **operator==** (const [bin\\_search\\_tree\\_const\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Refer-  
ence, Const\_Reference, !Is\_Forward\_Iterator, \_Alloc > &other) const

#### Public Attributes

- Node\_Pointer **m\_p\_nd**

#### Protected Types

- typedef [bin\\_search\\_tree\\_const\\_it\\_<](#) Node\_Pointer, Value\_Type, Pointer, Const\_Pointer, Reference, Const\_↵ Reference, Is\_Forward\_Iterator, \_Alloc > **base\_it\_type**

#### Protected Member Functions

- void **dec** (false\_type)
- void **dec** (true\_type)
- void **inc** (false\_type)
- void **inc** (true\_type)

#### 4.315.1 Detailed Description

```
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename
Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_↵
_Foward_Iterator, _Alloc >
```

Iterator.

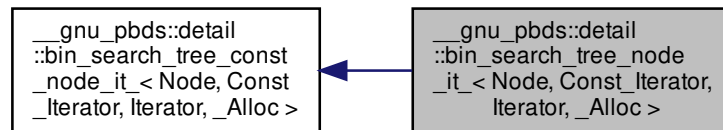
Definition at line 282 of file point\_iterators.hpp.

The documentation for this class was generated from the following file:

- [point\\_iterators.hpp](#)

#### 4.316 `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >`:



##### Public Types

- typedef Iterator [const\\_reference](#)
- typedef [trivial\\_iterator\\_difference\\_type](#) [difference\\_type](#)
- typedef [trivial\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef [rebind\\_traits< \\_Alloc, metadata\\_type >::const\\_reference](#) [metadata\\_const\\_reference](#)
- typedef Node::metadata\_type [metadata\\_type](#)
- typedef Iterator [reference](#)
- typedef Iterator [value\\_type](#)

##### Public Member Functions

- [bin\\_search\\_tree\\_node\\_it\\_](#) (const node\_pointer p\_nd=0)
- [bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_l\\_child](#) () const
- [metadata\\_const\\_reference](#) [get\\_metadata](#) () const
- [bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) [get\\_r\\_child](#) () const
- bool [operator!=](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const
- Iterator [operator\\*](#) () const
- bool [operator==](#) (const [bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#) &other) const

##### Public Attributes

- node\_pointer [m\\_p\\_nd](#)

##### 4.316.1 Detailed Description

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >
```

Node iterator.

Definition at line 132 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.316.2 Member Typedef Documentation

##### 4.316.2.1 `const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↵
_Alloc >::const_reference
```

Iterator's `__const` reference type.

Definition at line 146 of file `bin_search_tree_/node_iterators.hpp`.

##### 4.316.2.2 `difference_type`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type [inherited]
```

Difference type.

Definition at line 68 of file `bin_search_tree_/node_iterators.hpp`.

##### 4.316.2.3 `iterator_category`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↵
Iterator, Iterator, _Alloc >::iterator_category [inherited]
```

Category.

Definition at line 65 of file `bin_search_tree_/node_iterators.hpp`.

##### 4.316.2.4 `metadata_const_reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef rebind_traits<_Alloc, metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
```

Const metadata reference type.

Definition at line 84 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.316.2.5 `metadata_type`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_
Iterator, Iterator, _Alloc >::metadata_type [inherited]
```

Metadata type.

Definition at line 80 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.316.2.6 `reference`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _
Alloc >::reference
```

Iterator's reference type.

Definition at line 143 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.316.2.7 `value_type`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _
Alloc >::value_type
```

Iterator's value type.

Definition at line 140 of file `bin_search_tree_/node_iterators.hpp`.

### 4.316.3 Member Function Documentation

#### 4.316.3.1 `get_l_child()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
```

Returns the node iterator associated with the left node.

Definition at line 160 of file `bin_search_tree_/node_iterators.hpp`.

#### 4.316.3.2 get\_metadata()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline], [inherited]
```

Metadata access.

Definition at line 98 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.316.3.3 get\_r\_child()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bin_search_tree_node_it_<Node, Const_Iterator, Iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
```

Returns the node iterator associated with the right node.

Definition at line 168 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.316.3.4 operator!=(())

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline], [inherited]
```

Compares (negatively) to a different iterator object.

Definition at line 118 of file bin\_search\_tree\_/node\_iterators.hpp.

#### 4.316.3.5 operator\*()

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >↵
::operator* () const [inline]
```

Access.

Definition at line 155 of file bin\_search\_tree\_/node\_iterators.hpp.

4.316.3.6 `operator==()`

```
template<typename Node , class Const_Iterator , class Iterator , typename _Alloc >
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
 const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
 other) const [inline], [inherited]
```

Compares to a different iterator object.

Definition at line 113 of file `bin_search_tree_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [bin\\_search\\_tree\\_/node\\_iterators.hpp](#)

4.317 `__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >`

## Struct Template Reference

## Public Types

- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `Node` **node**
- typedef `bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_↵_update_pointer`
- typedef `bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

## 4.317.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename
_Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
```

Binary search tree traits, primary template.

Definition at line 63 of file `bin_search_tree_/traits.hpp`.

#### 4.317.2 Member Typedef Documentation

##### 4.317.2.1 node\_const\_iterator

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 128 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

#### 4.318 [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits](#)< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc > Struct Template Reference

##### Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#)< typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef Node **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#)< Node, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > [node\\_const\\_iterator](#)
- typedef [node\\_const\\_iterator](#) **node\_iterator**
- typedef Node\_Update< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#)< [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#)< typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef [point\\_const\\_iterator](#) **point\_iterator**
- typedef [const\\_reverse\\_iterator](#) **reverse\_iterator**

##### 4.318.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class
Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
```

Specialization.

Definition at line 166 of file `bin_search_tree_/traits.hpp`.



## 4.318.2 Member Typedef Documentation

## 4.318.2.1 node\_const\_iterator

```
template<typename Key , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node , typename _Alloc >
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::node_const_iterator
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

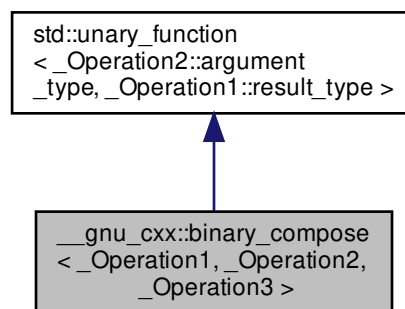
Definition at line 212 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [bin\\_search\\_tree\\_/traits.hpp](#)

## 4.319 \_\_gnu\_cxx::binary\_compose&lt; \_Operation1, \_Operation2, \_Operation3 &gt; Class Template Reference

Inheritance diagram for `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`:



## Public Types

- typedef `_Operation2::argument_type` [argument\\_type](#)
- typedef `_Operation1::result_type` [result\\_type](#)

### Public Member Functions

- **binary\_compose** (const \_Operation1 &\_\_x, const \_Operation2 &\_\_y, const \_Operation3 &\_\_z)
- \_Operation1::result\_type **operator()** (const typename \_Operation2::argument\_type &\_\_x) const

### Protected Attributes

- \_Operation1 **\_M\_fn1**
- \_Operation2 **\_M\_fn2**
- \_Operation3 **\_M\_fn3**

#### 4.319.1 Detailed Description

```
template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
```

An [SGI extension](#) .

Definition at line 142 of file ext/functional.

#### 4.319.2 Member Typedef Documentation

##### 4.319.2.1 argument\_type

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type , _Operation1↵
::result_type >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

##### 4.319.2.2 result\_type

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type , _Operation1↵
::result_type >::result_type [inherited]
```

result\_type is the return type

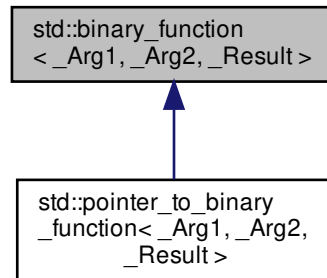
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 4.320 std::binary\_function< \_Arg1, \_Arg2, \_Result > Struct Template Reference

Inheritance diagram for std::binary\_function< \_Arg1, \_Arg2, \_Result >:



### Public Types

- typedef \_Arg1 [first\\_argument\\_type](#)
- typedef \_Result [result\\_type](#)
- typedef \_Arg2 [second\\_argument\\_type](#)

#### 4.320.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >
```

This is one of the [functor base classes](#).

Definition at line 118 of file `stl_function.h`.

#### 4.320.2 Member Typedef Documentation

##### 4.320.2.1 first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

#### 4.320.2.2 result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.320.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
```

second\_argument\_type is the type of the second argument

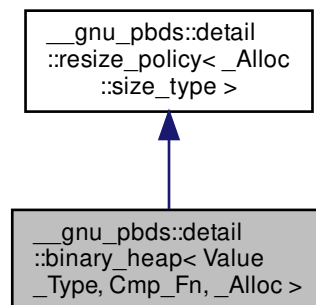
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.321 \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `cond_dealtor< value_type, _Alloc >` **cond\_dealtor\_t**
- typedef `binary_heap_const_iterator< value_type, entry, simple_value, _Alloc >` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `__conditional_type< simple_value, value_type, pointer >::__type` **entry**
- typedef `rebind_traits< _Alloc, entry >::allocator_type` **entry\_allocator**
- typedef `entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type` **entry\_cmp**
- typedef `rebind_traits< _Alloc, entry >::pointer` **entry\_pointer**
- typedef `const_iterator` **iterator**
- typedef `binary_heap_point_const_iterator< value_type, entry, simple_value, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `__gnu_pbds::detail::resize_policy< typename _Alloc::size_type >` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binary\_heap** (const `cmp_fn` &)
- **binary\_heap** (const `binary_heap` &)
- **iterator** **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- void **erase\_at** (`entry_pointer`, `size_type`, `false_type`)
- void **erase\_at** (`entry_pointer`, `size_type`, `true_type`)
- template<typename `Pred` >  
  `size_type` **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- bool **grow\_needed** (`size_type`) const
- void **join** (`binary_heap` &)
- `size_type` **max\_size** () const
- void **modify** (`point_iterator`, `const_reference`)
- void **notify\_arbitrary** (`size_type`)
- void **notify\_grow\_resize** ()
- void **notify\_shrink\_resize** ()
- void **pop** ()

- [point\\_iterator](#) **push** (const\_reference)
- bool **resize\_needed\_for\_grow** (size\_type) const
- bool **resize\_needed\_for\_shrink** (size\_type) const
- bool **shrink\_needed** (size\_type) const
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, [binary\\_heap](#) &)
- void **swap** ([resize\\_policy](#)<\_Alloc::size\_type > &)
- void **swap** ([binary\\_heap](#) &)
- const\_reference **top** () const

#### Static Public Attributes

- static const \_Alloc::size\_type **min\_size**

#### Protected Member Functions

- template<typename It >  
void **copy\_from\_range** (It, It)

#### 4.321.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binary heaps composed of resize and compare policies.

Based on CLRS.

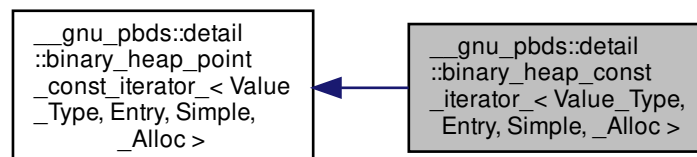
Definition at line 84 of file [binary\\_heap.hpp](#).

The documentation for this class was generated from the following file:

- [binary\\_heap.hpp](#)

#### 4.322 \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >:



## Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

## Public Member Functions

- `binary_heap_const_iterator_` (`entry_pointer p_e`)
- `binary_heap_const_iterator_` ()
- `binary_heap_const_iterator_` (`const binary_heap_const_iterator_ &other`)
- `bool operator!=` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator!=` (`const binary_heap_point_const_iterator_ &other`) `const`
- `const_reference operator*` () `const`
- `binary_heap_const_iterator_ & operator++` ()
- `binary_heap_const_iterator_ operator++` (`int`)
- `const_pointer operator->` () `const`
- `bool operator==` (`const binary_heap_const_iterator_ &other`) `const`
- `bool operator==` (`const binary_heap_point_const_iterator_ &other`) `const`

## Public Attributes

- `entry_pointer m_p_e`

### 4.322.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_ < Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `binary_heap_/const_iterator.hpp`.

### 4.322.2 Member Typedef Documentation

#### 4.322.2.1 `const_pointer`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 80 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.2.2 `const_reference`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 86 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.2.3 `difference_type`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type
```

Difference type.

Definition at line 71 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.2.4 `iterator_category`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```

Category.

Definition at line 68 of file `binary_heap_/const_iterator.hpp`.



#### 4.322.2.5 pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 77 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.2.6 reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::reference
```

Iterator's reference type.

Definition at line 83 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.2.7 value\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```

Iterator's value type.

Definition at line 74 of file `binary_heap_/const_iterator.hpp`.

### 4.322.3 Constructor & Destructor Documentation

#### 4.322.3.1 `binary_heap_const_iterator_()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_
() [inline]
```

Default constructor.

Definition at line 94 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.3.2 `binary_heap_const_iterator_()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_heap_const_iterator_
(
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```

Copy constructor.

Definition at line 99 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.4 Member Function Documentation

##### 4.322.4.1 `operator!=()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=
(
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 110 of file `binary_heap_/const_iterator.hpp`.

##### 4.322.4.2 `operator!=()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!= (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 122 of file `binary_heap_/point_const_iterator.hpp`.

##### 4.322.4.3 `operator*()`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline], [inherited]
```

Access.

Definition at line 109 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.322.4.4 `operator->()`

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↵
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

Definition at line 101 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.322.4.5 `operator==()` [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==
(
 const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const [inline]
```

Compares content to a different iterator object.

Definition at line 105 of file `binary_heap_/const_iterator.hpp`.

#### 4.322.4.6 `operator==()` [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator== (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline], [inherited]
```

Compares content to a different iterator object.

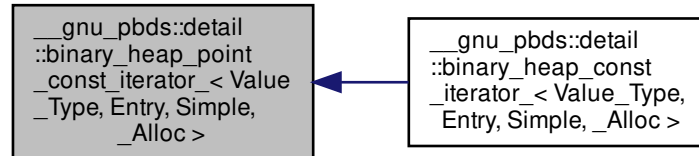
Definition at line 117 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/const\\_iterator.hpp](#)

#### 4.323 `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >`:



##### Public Types

- typedef `rebind_traits< _Alloc, value_type >::const_pointer` `const_pointer`
- typedef `rebind_traits< _Alloc, value_type >::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `rebind_traits< _Alloc, value_type >::pointer` `pointer`
- typedef `rebind_traits< _Alloc, value_type >::reference` `reference`
- typedef `Value_Type` `value_type`

##### Public Member Functions

- `binary_heap_point_const_iterator_` (`entry_pointer p_e`)
- `binary_heap_point_const_iterator_` ()
- `binary_heap_point_const_iterator_` (`const binary_heap_point_const_iterator_ &other`)
- `bool operator!=` (`const binary_heap_point_const_iterator_ &other`) `const`
- `const_reference operator*` () `const`
- `const_pointer operator->` () `const`
- `bool operator==` (`const binary_heap_point_const_iterator_ &other`) `const`

##### Public Attributes

- `entry_pointer m_p_e`

##### Protected Types

- typedef `rebind_traits< _Alloc, Entry >::pointer` `entry_pointer`

#### 4.323.1 Detailed Description

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >
```

Const point-type iterator.

Definition at line 55 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.323.2 Member Typedef Documentation

##### 4.323.2.1 const\_pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_
Value_Type, Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 75 of file `binary_heap_/point_const_iterator.hpp`.

##### 4.323.2.2 const\_reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_
Value_Type, Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 83 of file `binary_heap_/point_const_iterator.hpp`.

##### 4.323.2.3 difference\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_
Value_Type, Entry, Simple, _Alloc >::difference_type
```

Difference type.

Definition at line 65 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.323.2.4 iterator\_category

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```

Category.

Definition at line 62 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.2.5 pointer

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 71 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.2.6 reference

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference
```

Iterator's reference type.

Definition at line 79 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.2.7 value\_type

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```

Iterator's value type.

Definition at line 68 of file binary\_heap\_/point\_const\_iterator.hpp.

### 4.323.3 Constructor & Destructor Documentation

#### 4.323.3.1 binary\_heap\_point\_const\_iterator\_() [1/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ () [inline]
```

Default constructor.

Definition at line 91 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.3.2 binary\_heap\_point\_const\_iterator\_() [2/2]

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::binary_heap_point_const_iterator_ (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) [inline]
```

Copy constructor.

Definition at line 95 of file binary\_heap\_/point\_const\_iterator.hpp.

### 4.323.4 Member Function Documentation

#### 4.323.4.1 operator!=(())

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↵
::operator!=(
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 122 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.4.2 operator\*()

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const [inline]
```

Access.

Definition at line 109 of file binary\_heap\_/point\_const\_iterator.hpp.

#### 4.323.4.3 operator->()

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↔
_Alloc >::operator-> () const [inline]
```

Access.

Definition at line 101 of file `binary_heap_/point_const_iterator.hpp`.

#### 4.323.4.4 operator==()

```
template<typename Value_Type , typename Entry , bool Simple, typename _Alloc >
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↔
::operator== (
 const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other
) const [inline]
```

Compares content to a different iterator object.

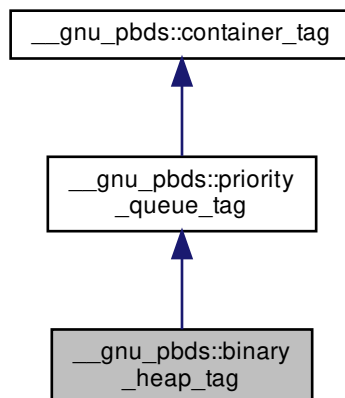
Definition at line 117 of file `binary_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [binary\\_heap\\_/point\\_const\\_iterator.hpp](#)

### 4.324 \_\_gnu\_pbds::binary\_heap\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::binary_heap_tag`:





## 4.324.1 Detailed Description

Binary-heap (array-based).

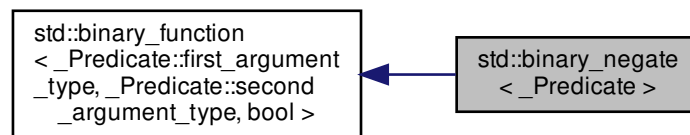
Definition at line 183 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.325 `std::binary_negate<_Predicate>` Class Template Reference

Inheritance diagram for `std::binary_negate<_Predicate>`:



## Public Types

- typedef `_Predicate::first_argument_type` [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef `_Predicate::second_argument_type` [second\\_argument\\_type](#)

## Public Member Functions

- constexpr **binary\_negate** (const `_Predicate` &\_\_x)
- constexpr bool **operator()** (const typename `_Predicate::first_argument_type` &\_\_x, const typename `_Predicate::second_argument_type` &\_\_y) const

## Protected Attributes

- `_Predicate` **\_M\_pred**

#### 4.325.1 Detailed Description

```
template<typename _Predicate>
class std::binary_negate<_Predicate >
```

One of the [negation functors](#).

Definition at line 1029 of file `stl_function.h`.

#### 4.325.2 Member Typedef Documentation

##### 4.325.2.1 first\_argument\_type

```
typedef _Predicate::first_argument_type std::binary_function<_Predicate::first_argument_type , ↔
_Predicate::second_argument_type , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.325.2.2 result\_type

```
typedef bool std::binary_function<_Predicate::first_argument_type , _Predicate::second_argument↔
_type , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.325.2.3 second\_argument\_type

```
typedef _Predicate::second_argument_type std::binary_function<_Predicate::first_argument_type ,
_Predicate::second_argument_type , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

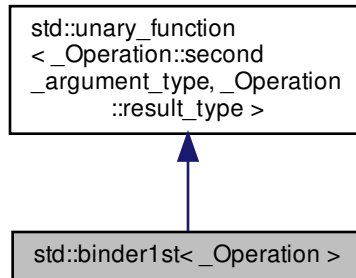
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.326 std::binder1st&lt; \_Operation &gt; Class Template Reference

Inheritance diagram for std::binder1st< \_Operation >:



## Public Types

- typedef `_Operation::second_argument_type` [argument\\_type](#)
- typedef `_Operation::result_type` [result\\_type](#)

## Public Member Functions

- **binder1st** (const `_Operation` &\_\_x, const typename `_Operation::first_argument_type` &\_\_y)
- `_Operation::result_type` **operator()** (const typename `_Operation::second_argument_type` &\_\_x) const
- `_Operation::result_type` **operator()** (typename `_Operation::second_argument_type` &\_\_x) const

## Protected Attributes

- `_Operation` **op**
- `_Operation::first_argument_type` **value**

## 4.326.1 Detailed Description

```
template<typename _Operation>
class std::binder1st< _Operation >
```

One of the [binder functors](#).

Definition at line 108 of file `binders.h`.

#### 4.326.2 Member Typedef Documentation

##### 4.326.2.1 `argument_type`

```
typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type ,
_Operation::result_type >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 4.326.2.2 `result_type`

```
typedef _Operation::result_type std::unary_function< _Operation::second_argument_type , _Operation↔
::result_type >::result_type [inherited]
```

`result_type` is the return type

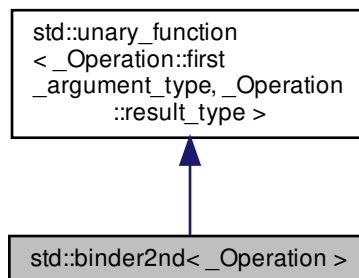
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

#### 4.327 `std::binder2nd< _Operation >` Class Template Reference

Inheritance diagram for `std::binder2nd< _Operation >`:



### Public Types

- typedef \_Operation::first\_argument\_type [argument\\_type](#)
- typedef \_Operation::result\_type [result\\_type](#)

### Public Member Functions

- **binder2nd** (const \_Operation &\_\_x, const typename \_Operation::second\_argument\_type &\_\_y)
- \_Operation::result\_type **operator()** (const typename \_Operation::first\_argument\_type &\_\_x) const
- \_Operation::result\_type **operator()** (typename \_Operation::first\_argument\_type &\_\_x) const

### Protected Attributes

- \_Operation **op**
- \_Operation::second\_argument\_type **value**

#### 4.327.1 Detailed Description

```
template<typename _Operation>
class std::binder2nd< _Operation >
```

One of the [binder functors](#).

Definition at line 143 of file binders.h.

#### 4.327.2 Member Typedef Documentation

##### 4.327.2.1 argument\_type

```
typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type , _↵
Operation::result_type >::argument_type [inherited]
```

[argument\\_type](#) is the type of the argument

Definition at line 108 of file stl\_function.h.

#### 4.327.2.2 `result_type`

```
typedef _Operation::result_type std::unary_function< _Operation::first_argument_type , _Operation↵
::result_type >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [binders.h](#)

### 4.328 `std::binomial_distribution< _IntType >` Class Template Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_IntType` [result\\_type](#)

#### Public Member Functions

- **`binomial_distribution`** (`_IntType` \_\_t, double \_\_p=0.5)
- **`binomial_distribution`** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **`generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **`generate`** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
void **`generate`** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **`max`** () const
- [result\\_type](#) **`min`** () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **`operator()`** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- double **`p`** () const
- [param\\_type](#) **`param`** () const
- void **`param`** (const [param\\_type](#) &\_\_param)
- void **`reset`** ()
- `_IntType` **`t`** () const

## Friends

- template<typename \_IntType1 , typename \_CharT , typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const std::binomial\_distribution< \_IntType1 > &\_\_x)
- bool operator== (const binomial\_distribution &\_\_d1, const binomial\_distribution &\_\_d2)
- template<typename \_IntType1 , typename \_CharT , typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is, std::binomial\_distribution< \_IntType1 > &\_\_x)

## 4.328.1 Detailed Description

```
template<typename _IntType = int>
class std::binomial_distribution< _IntType >
```

A discrete binomial random number distribution.

The formula for the binomial probability density function is  $p(i|t, p) = \binom{t}{i} p^i (1-p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 3738 of file random.h.

## 4.328.2 Member Typedef Documentation

## 4.328.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 3741 of file random.h.

## 4.328.3 Member Function Documentation

## 4.328.3.1 max()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3857 of file random.h.

#### 4.328.3.2 min()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3850 of file random.h.

#### 4.328.3.3 operator() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::binomial_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 3865 of file random.h.

#### 4.328.3.4 operator() [2/2]

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

A rejection algorithm when  $t * p \geq 8$  and a simple waiting time method - the second in the referenced book - otherwise.  
NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).

Definition at line 1524 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits< _Tp >::epsilon()`, `std::log()`, and `std::numeric_limits< _Tp >::max()`.

#### 4.328.3.5 p()

```
template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p () const [inline]
```

Returns the distribution `p` parameter.

Definition at line 3828 of file random.h.



**4.328.3.6 param()** [1/2]

```
template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3835 of file random.h.

**4.328.3.7 param()** [2/2]

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3843 of file random.h.

**4.328.3.8 reset()**

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Definition at line 3814 of file random.h.

References std::normal\_distribution< \_RealType >::reset().

**4.328.3.9 t()**

```
template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t () const [inline]
```

Returns the distribution `t` parameter.

Definition at line 3821 of file random.h.

#### 4.328.4 Friends And Related Function Documentation

##### 4.328.4.1 operator<<

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::binomial_distribution< _IntType1 > & __x) [friend]
```

Inserts a binomial\_distribution random number distribution \_\_x into the output stream \_\_os.

##### Parameters

|      |                                                     |
|------|-----------------------------------------------------|
| __os | An output stream.                                   |
| __x  | A binomial_distribution random number distribution. |

##### Returns

The output stream with the state of \_\_x inserted or in an error state.

##### 4.328.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
 const binomial_distribution< _IntType > & __d1,
 const binomial_distribution< _IntType > & __d2) [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3901 of file random.h.

##### 4.328.4.3 operator>>

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::binomial_distribution< _IntType1 > & __x) [friend]
```

Extracts a binomial\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| <code>__is</code> | An input stream.                                        |
| <code>__x</code>  | A binomial_distribution random number generator engine. |

## Returns

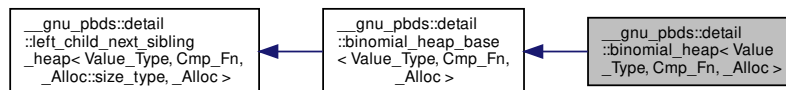
The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.329 `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `base_type::allocator_type` **allocator\_type**
- typedef `base_type::cmp_fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **binomial\_heap** (const Cmp\_Fn &)
- **binomial\_heap** (const [binomial\\_heap](#) &)
- [iterator](#) **begin** ()
- [const\\_iterator](#) **begin** () const
- void **clear** ()
- bool **empty** () const
- [iterator](#) **end** ()
- [const\\_iterator](#) **end** () const
- void **erase** ([point\\_iterator](#))
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **join** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- size\_type **max\_size** () const
- void **modify** ([point\\_iterator](#), const\_reference)
- void **pop** ()
- [point\\_iterator](#) **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap** ([left\\_child\\_next\\_sibling\\_heap](#)< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference **top** () const

## Protected Types

- typedef [base\\_type::node](#) **node**
- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef \_Alloc::size\_type **node\_metadata**
- typedef [std::pair](#)< node\_pointer, node\_pointer > **node\_pointer\_pair**

## Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap** ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

#### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

#### Protected Attributes

- node\_pointer **m\_p\_max**
- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 4.329.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Binomial heap.

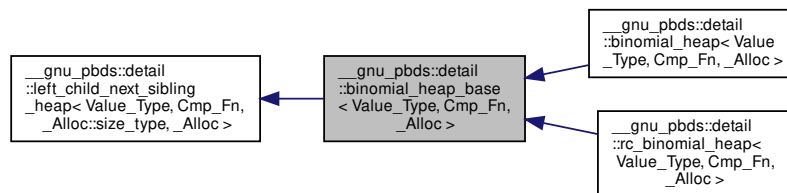
Definition at line 68 of file `binomial_heap_.hpp`.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_.hpp](#)

#### 4.330 `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >`:



### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- `iterator` **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- size\_type **max\_size** () const
- void **modify** (`point_iterator`, const\_reference)
- void **pop** ()
- `point_iterator` **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, `binomial_heap_base`< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap** (`left_child_next_sibling_heap`< `Value_Type`, `Cmp_Fn`, `_Alloc::size_type`, `_Alloc` > &)
- const\_reference **top** () const

### Protected Types

- typedef `base_type::node` **node**
- typedef `alloc_traits::allocator_type` **node\_allocator**
- typedef `base_type::node_const_pointer` **node\_const\_pointer**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `base_type::node_pointer` **node\_pointer**
- typedef `std::pair`< `node_pointer`, `node_pointer` > **node\_pointer\_pair**

### Protected Member Functions

- `binomial_heap_base` (const Cmp\_Fn &)
- `binomial_heap_base` (const [binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `actual_erase_node` (node\_pointer)
- void `bubble_to_top` (node\_pointer)
- void `clear_imp` (node\_pointer)
- template<typename It >  
void `copy_from_range` (It, It)
- void `find_max` ()
- node\_pointer `get_new_node_for_insert` (const\_reference)
- node\_pointer `prune` (Pred)
- void `swap` ([binomial\\_heap\\_base](#)< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap_with_parent` (node\_pointer, node\_pointer)
- void `to_linked_list` ()
- void `value_swap` ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void `make_child_of` (node\_pointer, node\_pointer)
- static node\_pointer `parent` (node\_pointer)

### Protected Attributes

- node\_pointer `m_p_max`
- node\_pointer `m_p_root`
- size\_type `m_size`

#### 4.330.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >
```

Base class for binomial heap.

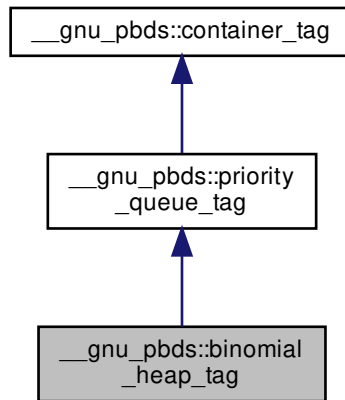
Definition at line 77 of file `binomial_heap_base_.hpp`.

The documentation for this class was generated from the following file:

- [binomial\\_heap\\_base\\_.hpp](#)

### 4.331 `__gnu_pbds::binomial_heap_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::binomial_heap_tag`:



#### 4.331.1 Detailed Description

Binomial-heap.

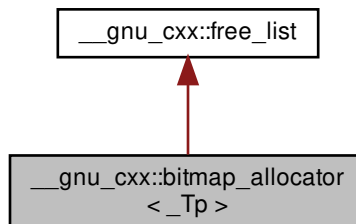
Definition at line 177 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.332 `__gnu_cxx::bitmap_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::bitmap_allocator<_Tp>`:





## Public Types

- `typedef free_list::__mutex_type __mutex_type`
- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true\_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- **bitmap\_allocator** (const [bitmap\\_allocator](#) &) noexcept
- `template<typename _Tp1 >`  
**bitmap\_allocator** (const [bitmap\\_allocator](#)<\_Tp1> &) noexcept
- `pointer \_M\_allocate\_single\_object ()`
- `void \_M\_deallocate\_single\_object (pointer __p) throw ()`
- `pointer address (reference __r) const noexcept`
- `const_pointer address (const_reference __r) const noexcept`
- `pointer allocate (size_type __n)`
- `pointer allocate (size_type __n, typename bitmap\_allocator< void >::const_pointer)`
- `template<typename _Up, typename... _Args>`  
`void construct (_Up *__p, _Args &&... __args)`
- `void deallocate (pointer __p, size_type __n) throw ()`
- `template<typename _Up >`  
`void destroy (_Up *__p)`
- `size_type max_size () const noexcept`

## Private Types

- `typedef vector_type::iterator iterator`
- `typedef \_\_detail::\_\_mini\_vector< value_type > vector_type`

## Private Member Functions

- `void \_M\_clear ()`
- `std::size_t * \_M\_get (std::size_t __sz)`
- `void \_M\_insert (std::size_t *__addr) throw ()`

## 4.332.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::bitmap_allocator<_Tp>
```

Bitmap Allocator, primary template.

Definition at line 658 of file `bitmap_allocator.h`.

#### 4.332.2 Member Function Documentation

##### 4.332.2.1 `_M_allocate_single_object()`

```
template<typename _Tp >
pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object () [inline]
```

Allocates memory for a single object of size `sizeof(_Tp)`.

##### Exceptions

|                              |                                |
|------------------------------|--------------------------------|
| <code>std::bad_alloc.</code> | If memory cannot be allocated. |
|------------------------------|--------------------------------|

Complexity: Worst case complexity is  $O(N)$ , but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of  $O(1)$ ! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

Definition at line 823 of file `bitmap_allocator.h`.

References `__gnu_cxx::__detail::__bit_allocate()`, `__gnu_cxx::__detail::__num_bitmaps()`, and `__gnu_cxx::_Bit_scan←_forward()`.

##### 4.332.2.2 `_M_deallocate_single_object()`

```
template<typename _Tp >
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
 pointer __p) throw () [inline]
```

Deallocates memory that belongs to a single object of size `sizeof(_Tp)`.

Complexity:  $O(\lg(N))$ , but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in  $O(1)$  time by the `deallocate` function.

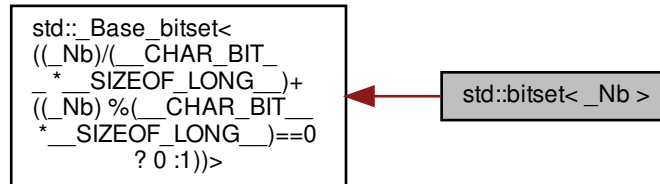
Definition at line 914 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 4.333 std::bitset&lt; \_Nb &gt; Class Template Reference

Inheritance diagram for std::bitset< \_Nb >:



## Classes

- class [reference](#)

## Public Member Functions

- constexpr [bitset](#) () noexcept
- constexpr [bitset](#) (unsigned long long \_\_val) noexcept
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position=0)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
[bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_position, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT >  
[bitset](#) (const \_CharT \* \_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- size\_t [\\_Find\\_first](#) () const noexcept
- size\_t [\\_Find\\_next](#) (size\_t \_\_prev) const noexcept
- template<class \_CharT, class \_Traits >  
void [\\_M\\_copy\\_from\\_ptr](#) (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT, \_CharT)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero, \_CharT \_\_one)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_from\\_string](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s, size\_t \_\_pos, size\_t \_\_n)
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &, \_CharT, \_CharT) const
- template<class \_CharT, class \_Traits, class \_Alloc >  
void [\\_M\\_copy\\_to\\_string](#) ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s) const
- bool [all](#) () const noexcept

- `bool any () const noexcept`
  - `size_t count () const noexcept`
  - `bitset<_Nb> & flip () noexcept`
  - `bitset<_Nb> & flip (size_t __position)`
  - `bool none () const noexcept`
  - `bitset<_Nb> operator~ () const noexcept`
  - `bitset<_Nb> & reset () noexcept`
  - `bitset<_Nb> & reset (size_t __position)`
  - `bitset<_Nb> & set () noexcept`
  - `bitset<_Nb> & set (size_t __position, bool __val=true)`
  - `constexpr size_t size () const noexcept`
  - `bool test (size_t __position) const`
  - `template<class _CharT, class _Traits, class _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc> to_string () const`
  - `template<class _CharT, class _Traits, class _Alloc>  
std::basic_string<_CharT, _Traits, _Alloc> to_string (_CharT __zero, _CharT __one=_CharT('1')) const`
  - `template<class _CharT, class _Traits>  
std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string () const`
  - `template<class _CharT, class _Traits>  
std::basic_string<_CharT, _Traits, std::allocator<_CharT>> to_string (_CharT __zero, _CharT __one=_↵  
CharT('1')) const`
  - `template<class _CharT>  
std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string () const`
  - `template<class _CharT>  
std::basic_string<_CharT, std::char_traits<_CharT>, std::allocator<_CharT>> to_string (_CharT __zero,  
_CharT __one=_CharT('1')) const`
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string () const`
  - `std::basic_string<char, std::char_traits<char>, std::allocator<char>> to_string (char __zero, char __↵  
one='1') const`
  - `unsigned long long to_ullong () const`
  - `unsigned long to_ulong () const`
- 
- `bitset<_Nb> & operator&= (const bitset<_Nb> &__rhs) noexcept`
  - `bitset<_Nb> & operator|= (const bitset<_Nb> &__rhs) noexcept`
  - `bitset<_Nb> & operator^= (const bitset<_Nb> &__rhs) noexcept`
- 
- `bitset<_Nb> & operator<<= (size_t __position) noexcept`
  - `bitset<_Nb> & operator>>= (size_t __position) noexcept`
- 
- `bitset<_Nb> & _Unchecked_set (size_t __pos) noexcept`
  - `bitset<_Nb> & _Unchecked_set (size_t __pos, int __val) noexcept`
  - `bitset<_Nb> & _Unchecked_reset (size_t __pos) noexcept`

- [bitset<\\_Nb> & \\_Unchecked\\_flip](#) (size\_t \_\_pos) noexcept
- constexpr bool [\\_Unchecked\\_test](#) (size\_t \_\_pos) const noexcept
- [reference operator\[\]](#) (size\_t \_\_position)
- constexpr bool [operator\[\]](#) (size\_t \_\_position) const
- bool [operator==](#) (const [bitset<\\_Nb>](#) & \_\_rhs) const noexcept
- bool [operator!=](#) (const [bitset<\\_Nb>](#) & \_\_rhs) const noexcept
- [bitset<\\_Nb> operator<<](#) (size\_t \_\_position) const noexcept
- [bitset<\\_Nb> operator>>](#) (size\_t \_\_position) const noexcept

#### Private Member Functions

- bool [\\_M\\_are\\_all](#) () const noexcept
- void [\\_M\\_do\\_and](#) (const [\\_Base\\_bitset<\\_Nw>](#) & \_\_x) noexcept
- size\_t [\\_M\\_do\\_count](#) () const noexcept
- size\_t [\\_M\\_do\\_find\\_first](#) (size\_t) const noexcept
- size\_t [\\_M\\_do\\_find\\_next](#) (size\_t, size\_t) const noexcept
- void [\\_M\\_do\\_flip](#) () noexcept
- void [\\_M\\_do\\_left\\_shift](#) (size\_t \_\_shift) noexcept
- void [\\_M\\_do\\_or](#) (const [\\_Base\\_bitset<\\_Nw>](#) & \_\_x) noexcept
- void [\\_M\\_do\\_reset](#) () noexcept
- void [\\_M\\_do\\_right\\_shift](#) (size\_t \_\_shift) noexcept
- void [\\_M\\_do\\_set](#) () noexcept
- unsigned long long [\\_M\\_do\\_to\\_ullong](#) () const
- unsigned long [\\_M\\_do\\_to\\_ulong](#) () const
- void [\\_M\\_do\\_xor](#) (const [\\_Base\\_bitset<\\_Nw>](#) & \_\_x) noexcept
- const \_WordT \* [\\_M\\_getdata](#) () const noexcept
- \_WordT & [\\_M\\_getword](#) (size\_t \_\_pos) noexcept
- constexpr \_WordT [\\_M\\_getword](#) (size\_t \_\_pos) const noexcept
- \_WordT & [\\_M\\_hiword](#) () noexcept
- constexpr \_WordT [\\_M\\_hiword](#) () const noexcept
- bool [\\_M\\_is\\_any](#) () const noexcept
- bool [\\_M\\_is\\_equal](#) (const [\\_Base\\_bitset<\\_Nw>](#) & \_\_x) const noexcept

#### Static Private Member Functions

- static constexpr \_WordT [\\_S\\_maskbit](#) (size\_t \_\_pos) noexcept
- static constexpr size\_t [\\_S\\_whichbit](#) (size\_t \_\_pos) noexcept
- static constexpr size\_t [\\_S\\_whichbyte](#) (size\_t \_\_pos) noexcept
- static constexpr size\_t [\\_S\\_whichword](#) (size\_t \_\_pos) noexcept

## Private Attributes

- `_WordT` `_M_w` [`_Nw`]

## Friends

- class **reference**
- struct **std::hash**< **bitset** >

### 4.333.1 Detailed Description

```
template<size_t _Nb>
class std::bitset<_Nb>
```

The `bitset` class represents a *fixed-size* sequence of bits.

(Note that `bitset` does *not* meet the formal requirements of a [container](#). Mainly, it lacks iterators.)

The template argument, *Nb*, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024\*1024").

In the general unoptimized case, storage is allocated in word-sized blocks. Let *B* be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage. *B* - *NbB* bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `bitset` as a *simple array of bits*, be aware that your mental picture is reversed: a `bitset` behaves the same way as bits in integers do, with the bit at index 0 in the *least significant* / *right-hand* position, and the bit at index *Nb*-1 in the *most significant* / *left-hand* position. Thus, unlike other containers, a `bitset`'s index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
 long a = 'a';
 bitset<10> b(a);
 cout << "b('a') is " << b << endl;
 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}
```

Also see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_containers.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html) for a description of extensions.

Most of the actual code isn't contained in `bitset<>` itself, but in the base class `_Base_bitset`. The base class works with whole words, not with individual bits. This allows us to specialize `_Base_bitset` for the important special case where the `bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `_Base_bitset` is a regular array, and is indexed as such. This is carefully encapsulated.

Definition at line 751 of file `bitset`.

## 4.333.2 Constructor &amp; Destructor Documentation

## 4.333.2.1 bitset() [1/5]

```
template<size_t _Nb>
constexpr std::bitset< _Nb >::bitset () [inline], [noexcept]
```

All bits set to zero.

Definition at line 869 of file `bitset`.

## 4.333.2.2 bitset() [2/5]

```
template<size_t _Nb>
constexpr std::bitset< _Nb >::bitset (
 unsigned long long __val) [inline], [noexcept]
```

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 874 of file `bitset`.

## 4.333.2.3 bitset() [3/5]

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::bitset< _Nb >::bitset (
 const std::basic_string< _CharT, _Traits, _Alloc > & __s,
 size_t __position = 0) [inline], [explicit]
```

Use a subset of a string.

## Parameters

|                         |                                                                            |
|-------------------------|----------------------------------------------------------------------------|
| <code>__s</code>        | A string of 0 and 1 characters.                                            |
| <code>__position</code> | Index of the first character in <code>__s</code> to use; defaults to zero. |

## Exceptions

|                                    |                                                                |
|------------------------------------|----------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>pos</code> is bigger the size of <code>__s</code> .   |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither 0 nor 1. |

Definition at line 893 of file `bitset`.

4.333.2.4 `bitset()` [4/5]

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::bitset< _Nb >::bitset (
 const std::basic_string< _CharT, _Traits, _Alloc > & __s,
 size_t __position,
 size_t __n) [inline]
```

Use a subset of a string.

## Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__s</code>        | A string of <i>0</i> and <i>1</i> characters.            |
| <code>__position</code> | Index of the first character in <code>__s</code> to use. |
| <code>__n</code>        | The number of characters to copy.                        |

## Exceptions

|                                    |                                                                               |
|------------------------------------|-------------------------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>__position</code> is bigger the size of <code>__s</code> .           |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither <i>0</i> nor <i>1</i> . |

Definition at line 914 of file `bitset`.

4.333.2.5 `bitset()` [5/5]

```
template<size_t _Nb>
template<typename _CharT >
std::bitset< _Nb >::bitset (
 const _CharT * __str,
 typename std::basic_string< _CharT >::size_type __n = std::basic_string<_CharT>::npos,
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1')) [inline], [explicit]
```

Construct from a character array.

## Parameters

|                     |                                                     |
|---------------------|-----------------------------------------------------|
| <code>__str</code>  | An array of characters <i>zero</i> and <i>one</i> . |
| <code>__n</code>    | The number of characters to use.                    |
| <code>__zero</code> | The character corresponding to the value 0.         |
| <code>__one</code>  | The character corresponding to the value 1.         |



**Exceptions**

|                                    |                                                                                                    |
|------------------------------------|----------------------------------------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither <code>__zero</code> nor <code>__one</code> . |
|------------------------------------|----------------------------------------------------------------------------------------------------|

Definition at line 946 of file `bitset`.

**4.333.3 Member Function Documentation****4.333.3.1 all()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::all () const [inline], [noexcept]
```

Tests whether all the bits are on.

**Returns**

True if all the bits are set.

Definition at line 1336 of file `bitset`.

**4.333.3.2 any()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::any () const [inline], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if at least one bit is set.

Definition at line 1344 of file `bitset`.

**4.333.3.3 count()**

```
template<size_t _Nb>
size_t std::bitset<_Nb>::count () const [inline], [noexcept]
```

Returns the number of bits which are set.

Definition at line 1295 of file `bitset`.

**4.333.3.4 flip()** [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::flip () [inline], [noexcept]
```

Toggles every bit to its opposite value.

Definition at line 1123 of file `bitset`.

**4.333.3.5 flip()** [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::flip (
 size_t __position) [inline]
```

Toggles a given bit to its opposite value.

**Parameters**

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

Definition at line 1136 of file `bitset`.

**4.333.3.6 none()**

```
template<size_t _Nb>
bool std::bitset<_Nb>::none () const [inline], [noexcept]
```

Tests whether any of the bits are on.

**Returns**

True if none of the bits are set.

Definition at line 1352 of file `bitset`.

## 4.333.3.7 operator!=(())

```
template<size_t _Nb>
bool std::bitset<_Nb>::operator!=(
 const bitset<_Nb> & __rhs) const [inline], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1311 of file `bitset`.

## 4.333.3.8 operator&amp;=()

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator&= (
 const bitset<_Nb> & __rhs) [inline], [noexcept]
```

Operations on bitsets.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__rhs</code> | A same-sized <code>bitset</code> . |
|--------------------|------------------------------------|

These should be self-explanatory.

Definition at line 972 of file `bitset`.

## 4.333.3.9 operator&lt;&lt;()

```
template<size_t _Nb>
bitset<_Nb> std::bitset<_Nb>::operator<< (
 size_t __position) const [inline], [noexcept]
```

Self-explanatory.

Definition at line 1358 of file `bitset`.

## 4.333.3.10 operator&lt;&lt;=()

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator<<= (
 size_t __position) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|

These should be self-explanatory.

Definition at line 1001 of file `bitset`.

**4.333.3.11 `operator==()`**

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator== (
 const bitset< _Nb > & __rhs) const [inline], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

Definition at line 1306 of file `bitset`.

**4.333.3.12 `operator>>()`**

```
template<size_t _Nb>
bitset<_Nb> std::bitset< _Nb >::operator>> (
 size_t __position) const [inline], [noexcept]
```

Self-explanatory.

Definition at line 1362 of file `bitset`.

**4.333.3.13 `operator>>=()`**

```
template<size_t _Nb>
bitset<_Nb>& std::bitset< _Nb >::operator>>= (
 size_t __position) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

|                         |                                |
|-------------------------|--------------------------------|
| <code>__position</code> | The number of places to shift. |
|-------------------------|--------------------------------|

These should be self-explanatory.

Definition at line 1014 of file `bitset`.

## 4.333.3.14 operator[]() [1/2]

```
template<size_t _Nb>
reference std::bitset<_Nb>::operator[] (
 size_t __position) [inline]
```

Array-indexing support.

## Parameters

|                         |                        |
|-------------------------|------------------------|
| <code>__position</code> | Index into the bitset. |
|-------------------------|------------------------|

## Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

## Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

\_GLIBCXX\_RESOLVE\_LIB\_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1163 of file bitset.

## 4.333.3.15 operator[]() [2/2]

```
template<size_t _Nb>
constexpr bool std::bitset<_Nb>::operator[] (
 size_t __position) const [inline]
```

Array-indexing support.

## Parameters

|                         |                        |
|-------------------------|------------------------|
| <code>__position</code> | Index into the bitset. |
|-------------------------|------------------------|

## Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

`__GLIBCXX_RESOLVE_LIB_DEFECTS` Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

Definition at line 1167 of file `bitset`.

**4.333.3.16 `operator^=()`**

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator^= (
 const bitset<_Nb> & __rhs) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

Definition at line 986 of file `bitset`.

**4.333.3.17 `operator" |=()`**

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::operator|= (
 const bitset<_Nb> & __rhs) [inline], [noexcept]
```

Operations on bitsets.

**Parameters**

|                    |                      |
|--------------------|----------------------|
| <code>__rhs</code> | A same-sized bitset. |
|--------------------|----------------------|

These should be self-explanatory.

Definition at line 979 of file `bitset`.

## 4.333.3.18 operator~()

```
template<size_t _Nb>
bitset<_Nb> std::bitset<_Nb>::operator~ () const [inline], [noexcept]
```

See the no-argument flip().

Definition at line 1144 of file `bitset`.

## 4.333.3.19 reset() [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::reset () [inline], [noexcept]
```

Sets every bit to false.

Definition at line 1099 of file `bitset`.

## 4.333.3.20 reset() [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::reset (
 size_t __position) [inline]
```

Sets a given bit to false.

## Parameters

|                         |                       |
|-------------------------|-----------------------|
| <code>__position</code> | The index of the bit. |
|-------------------------|-----------------------|

## Exceptions

|                                |                                                    |
|--------------------------------|----------------------------------------------------|
| <code>std::out_of_range</code> | If <code>pos</code> is bigger the size of the set. |
|--------------------------------|----------------------------------------------------|

Same as writing `set (pos, false)`.

Definition at line 1113 of file `bitset`.

## 4.333.3.21 set() [1/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::set () [inline], [noexcept]
```

Sets every bit to true.

Definition at line 1075 of file `bitset`.

#### 4.333.3.22 `set()` [2/2]

```
template<size_t _Nb>
bitset<_Nb>& std::bitset<_Nb>::set (
 size_t __position,
 bool __val = true) [inline]
```

Sets a given bit to a particular value.

##### Parameters

|                         |                                         |
|-------------------------|-----------------------------------------|
| <code>__position</code> | The index of the bit.                   |
| <code>__val</code>      | Either true or false, defaults to true. |

##### Exceptions

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

Definition at line 1089 of file `bitset`.

#### 4.333.3.23 `size()`

```
template<size_t _Nb>
constexpr size_t std::bitset<_Nb>::size () const [inline], [noexcept]
```

Returns the total number of bits.

Definition at line 1300 of file `bitset`.

#### 4.333.3.24 `test()`

```
template<size_t _Nb>
bool std::bitset<_Nb>::test (
 size_t __position) const [inline]
```

Tests the value of a bit.



**Parameters**

|                         |                     |
|-------------------------|---------------------|
| <code>__position</code> | The index of a bit. |
|-------------------------|---------------------|

**Returns**

The value at *pos*.

**Exceptions**

|                                |                                              |
|--------------------------------|----------------------------------------------|
| <code>std::out_of_range</code> | If <i>pos</i> is bigger the size of the set. |
|--------------------------------|----------------------------------------------|

Definition at line 1323 of file `bitset`.

**4.333.3.25 to\_string()**

```
template<size_t _Nb>
template<class _CharT , class _Traits , class _Alloc >
std::basic_string<_CharT, _Traits, _Alloc> std::bitset< _Nb >::to_string () const [inline]
```

Returns a character interpretation of the bitset.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 1197 of file `bitset`.

**4.333.3.26 to\_ulong()**

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong () const [inline]
```

Returns a numerical interpretation of the bitset.

**Returns**

The integral equivalent of the bits.

## Exceptions

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

Definition at line 1178 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)

#### 4.334 `std::__debug::bitset<_Nb>` Class Template Reference

Inherits `bitset<_Nb>`.

## Public Types

- typedef `_Base::reference` **reference**

## Public Member Functions

- constexpr **bitset** (unsigned long long \_\_val) noexcept
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos=0, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::npos)
- template<class \_CharT, class \_Traits, class \_Alloc >  
**bitset** (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_str, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_pos, typename [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc >::size\_type \_\_n, \_CharT \_\_zero, \_CharT \_\_one=\_CharT('1'))
- **bitset** (const [\\_Base](#) &\_\_x)
- template<typename \_CharT >  
**bitset** (const \_CharT \*\_\_str, typename [std::basic\\_string](#)< \_CharT >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- [\\_Base](#) & **\_M\_base** () noexcept
- const [\\_Base](#) & **\_M\_base** () const noexcept
- [bitset](#)<\_Nb> & **flip** () noexcept
- [bitset](#)<\_Nb> & **flip** (size\_t \_\_pos)
- bool **operator!=** (const [bitset](#)<\_Nb> &\_\_rhs) const noexcept
- [bitset](#)<\_Nb> & **operator&=** (const [bitset](#)<\_Nb> &\_\_rhs) noexcept
- [bitset](#)<\_Nb> **operator<<** (size\_t \_\_pos) const noexcept
- [bitset](#)<\_Nb> & **operator<<=** (size\_t \_\_pos) noexcept
- bool **operator==** (const [bitset](#)<\_Nb> &\_\_rhs) const noexcept
- [bitset](#)<\_Nb> **operator>>** (size\_t \_\_pos) const noexcept
- [bitset](#)<\_Nb> & **operator>>=** (size\_t \_\_pos) noexcept
- reference **operator[]** (size\_t \_\_pos)
- constexpr bool **operator[]** (size\_t \_\_pos) const
- [bitset](#)<\_Nb> & **operator^=** (const [bitset](#)<\_Nb> &\_\_rhs) noexcept

- [bitset](#)< \_Nb > & **operator**|= (const [bitset](#)< \_Nb > &\_\_rhs) noexcept
- [bitset](#)< \_Nb > **operator**~ () const noexcept
- [bitset](#)< \_Nb > & **reset** () noexcept
- [bitset](#)< \_Nb > & **reset** (size\_t \_\_pos)
- [bitset](#)< \_Nb > & **set** () noexcept
- [bitset](#)< \_Nb > & **set** (size\_t \_\_pos, bool \_\_val=true)
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** () const
- template<class \_CharT, class \_Traits, class \_Alloc >  
[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_CharT('1')) const
- template<typename \_CharT, typename \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<class \_CharT, class \_Traits >  
[std::basic\\_string](#)< \_CharT, \_Traits, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero, \_CharT \_\_one=\_↵  
\_CharT('1')) const
- template<typename \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** () const
- template<class \_CharT >  
[std::basic\\_string](#)< \_CharT, [std::char\\_traits](#)< \_CharT >, [std::allocator](#)< \_CharT > > **to\_string** (\_CharT \_\_zero,  
\_CharT \_\_one=\_CharT('1')) const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** () const
- [std::basic\\_string](#)< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char > > **to\_string** (char \_\_zero, char \_\_↵  
one='1') const

#### 4.334.1 Detailed Description

```
template<size_t _Nb>
class std::__debug::bitset< _Nb >
```

Class std::bitset with additional safety/checking/debug instrumentation.

Definition at line 44 of file debug/bitset.

The documentation for this class was generated from the following file:

- [debug/bitset](#)

## 4.335 std::tr2::bool\_set Class Reference

### Public Member Functions

- constexpr [bool\\_set](#) ()
- constexpr [bool\\_set](#) (bool \_\_t)
- bool **contains** ([bool\\_set](#) \_\_b) const
- bool **equals** ([bool\\_set](#) \_\_b) const
- bool **is\_emptyset** () const
- bool **is\_indeterminate** () const
- bool **is\_singleton** () const
- **operator bool** () const

### Static Public Member Functions

- static [bool\\_set](#) **emptyset** ()
- static [bool\\_set](#) **indeterminate** ()

### Friends

- [bool\\_set](#) **operator!** ([bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator&** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT , typename Traits >  
[std::basic\\_ostream](#)< CharT, Traits > & **operator**<< ([std::basic\\_ostream](#)< CharT, Traits > &\_\_out, [bool\\_set](#) \_\_b)
- [bool\\_set](#) **operator==** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- template<typename CharT , typename Traits >  
[std::basic\\_istream](#)< CharT, Traits > & **operator**>> ([std::basic\\_istream](#)< CharT, Traits > &\_\_in, [bool\\_set](#) &\_\_b)
- [bool\\_set](#) **operator^** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)
- [bool\\_set](#) **operator|** ([bool\\_set](#) \_\_s, [bool\\_set](#) \_\_t)

#### 4.335.1 Detailed Description

##### [bool\\_set](#)

See N2136, Bool\_set: multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.

The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

Definition at line 54 of file [bool\\_set](#).

#### 4.335.2 Constructor & Destructor Documentation

##### 4.335.2.1 [bool\\_set](#)() [1/2]

```
constexpr std::tr2::bool_set::bool_set () [inline]
```

Default constructor.

Definition at line 59 of file [bool\\_set](#).

##### 4.335.2.2 [bool\\_set](#)() [2/2]

```
constexpr std::tr2::bool_set::bool_set (
 bool __t) [inline]
```

Constructor from bool.

Definition at line 62 of file [bool\\_set](#).

### 4.335.3 Member Function Documentation

#### 4.335.3.1 equals()

```
bool std::tr2::bool_set::equals (
 bool_set __b) const [inline]
```

Return true if states are equal.

Definition at line 69 of file bool\_set.

#### 4.335.3.2 is\_emptyset()

```
bool std::tr2::bool_set::is_emptyset () const [inline]
```

Return true if this is empty.

Definition at line 73 of file bool\_set.

#### 4.335.3.3 is\_indeterminate()

```
bool std::tr2::bool_set::is_indeterminate () const [inline]
```

Return true if this is indeterminate.

Definition at line 77 of file bool\_set.

#### 4.335.3.4 is\_singleton()

```
bool std::tr2::bool_set::is_singleton () const [inline]
```

Return true if this is false or true (normal boolean).

Definition at line 81 of file bool\_set.

#### 4.335.3.5 operator bool()

```
std::tr2::bool_set::operator bool () const [inline]
```

Conversion to bool.

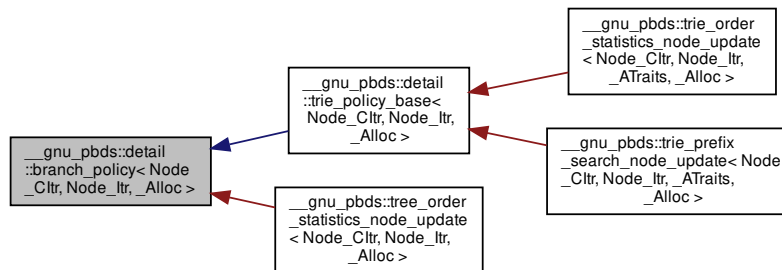
Definition at line 86 of file bool\_set.

The documentation for this class was generated from the following files:

- [bool\\_set](#)
- [bool\\_set.tcc](#)

#### 4.336 \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::branch\_policy< Node\_Cltr, Node\_Itr, \_Alloc >:



#### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `rebind_k::const_reference` **key\_const\_reference**
- typedef `value_type::first_type` **key\_type**
- typedef `remove_const< key_type >::type` **rckey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `rebind_traits< _Alloc, rckey_type >` **rebind\_k**
- typedef `rebind_traits< _Alloc, rcvalue_type >` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

#### Protected Member Functions

- virtual `it_type end ()=0`
- `it_type end_iterator () const`

#### Static Protected Member Functions

- static `key_const_reference` **extract\_key** (`const_reference` r\_val)

##### 4.336.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >
```

Primary template, base class for branch structure policies.

Definition at line 53 of file `branch_policy.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

#### 4.337 `__gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >` Struct Template Reference

#### Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Cltr::value_type` **it\_type**
- typedef `rebind_v::const_reference` **key\_const\_reference**
- typedef `value_type` **key\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `rebind_traits< _Alloc, rcvalue_type >` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

#### Protected Member Functions

- virtual `it_type` **end** () const =0
- `it_type` **end\_iterator** () const

#### Static Protected Member Functions

- static `key_const_reference` **extract\_key** (`const_reference` r\_val)

#### 4.337.1 Detailed Description

```
template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >
```

Specialization for const iterators.

Definition at line 89 of file branch\_policy.hpp.

The documentation for this struct was generated from the following file:

- [branch\\_policy.hpp](#)

#### 4.338 std::cauchy\_distribution<\_RealType> Class Template Reference

##### Classes

- struct [param\\_type](#)

##### Public Types

- typedef \_RealType [result\\_type](#)

##### Public Member Functions

- **cauchy\_distribution** (\_RealType \_\_a, \_RealType \_\_b=1.0)
- **cauchy\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()



## Friends

- bool `operator==` (const `cauchy_distribution` &\_\_d1, const `cauchy_distribution` &\_\_d2)

## 4.338.1 Detailed Description

```
template<typename _RealType = double>
class std::cauchy_distribution< _RealType >
```

A `cauchy_distribution` random number distribution.

The formula for the normal probability mass function is  $p(x|a,b) = (\pi b(1 + (\frac{x-a}{b})^2))^{-1}$

Definition at line 2854 of file `random.h`.

## 4.338.2 Member Typedef Documentation

## 4.338.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2857 of file `random.h`.

## 4.338.3 Member Function Documentation

## 4.338.3.1 a()

```
template<typename _RealType = double>
_RealType std::cauchy_distribution< _RealType >::a () const [inline]
```

Definition at line 2919 of file `random.h`.

#### 4.338.3.2 max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2952 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

#### 4.338.3.3 min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2945 of file random.h.

References `std::numeric_limits<_Tp>::lowest()`.

#### 4.338.3.4 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::cauchy_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 2960 of file random.h.

#### 4.338.3.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2930 of file random.h.

Referenced by `std::operator>>()`.

#### 4.338.3.6 param() [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2938 of file random.h.

4.338.3.7 `reset()`

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 2912 of file random.h.

## 4.338.4 Friends And Related Function Documentation

4.338.4.1 `operator==`

```
template<typename _RealType = double>
bool operator== (
 const cauchy_distribution< _RealType > & __d1,
 const cauchy_distribution< _RealType > & __d2) [friend]
```

Return true if two Cauchy distributions have the same parameters.

Definition at line 2995 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.339 `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`  
Class Template Reference

## Public Types

- enum { [external\\_load\\_access](#) }
- typedef `Size_Type` **size\_type**

### Public Member Functions

- `cc_hash_max_collision_check_resize_trigger` (float load=0.5)
- float `get_load` () const
- void `set_load` (float load)
- void `swap` (`cc_hash_max_collision_check_resize_trigger`< External\_Load\_Access, Size\_Type > &other)

### Protected Member Functions

- bool `is_grow_needed` (size\_type size, size\_type num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (size\_type num\_entries)
- void `notify_externally_resized` (size\_type new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (size\_type num\_entries)
- void `notify_resized` (size\_type new\_size)

#### 4.339.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

Definition at line 293 of file hash\_policy.hpp.

#### 4.339.2 Member Enumeration Documentation

##### 4.339.2.1 anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

## Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 298 of file `hash_policy.hpp`.

## 4.339.3 Constructor & Destructor Documentation

### 4.339.3.1 `cc_hash_max_collision_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type >
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_hash_max_collision_
(
 float load = 0.5)
```

Default constructor, or constructor taking `load`, a `__load` factor which it will attempt to maintain.

Definition at line 46 of file `hash_policy.hpp`.

## 4.339.4 Member Function Documentation

### 4.339.4.1 `get_load()`

```
template<bool External_Load_Access, typename Size_Type >
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::get_load () const [inline]
```

Returns the current load.

Definition at line 192 of file `hash_policy.hpp`.

### 4.339.4.2 `is_grow_needed()`

```
template<bool External_Load_Access, typename Size_Type >
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::is_grow_needed (
 size_type size,
 size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

Definition at line 135 of file `hash_policy.hpp`.

#### 4.339.4.3 is\_resize\_needed()

```
template<bool External_Load_Access, typename Size_Type >
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

Definition at line 129 of file hash\_policy.hpp.

#### 4.339.4.4 notify\_cleared()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_cleared () [protected]
```

Notifies the table was cleared.

Definition at line 123 of file hash\_policy.hpp.

#### 4.339.4.5 notify\_erase\_search\_collision()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 99 of file hash\_policy.hpp.

#### 4.339.4.6 notify\_erase\_search\_end()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

Definition at line 105 of file hash\_policy.hpp.

#### 4.339.4.7 notify\_erase\_search\_start()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erase_search_start () [inline], [protected]
```

Notifies an erase search started.

Definition at line 93 of file hash\_policy.hpp.

#### 4.339.4.8 notify\_erased()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_erased (
 size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

Definition at line 117 of file hash\_policy.hpp.

#### 4.339.4.9 notify\_externally\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_externally_resized (
 size_type new_size) [protected]
```

Notifies the table was resized externally.

Definition at line 174 of file hash\_policy.hpp.

#### 4.339.4.10 notify\_find\_search\_collision()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 63 of file hash\_policy.hpp.

#### 4.339.4.11 notify\_find\_search\_end()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

Definition at line 69 of file hash\_policy.hpp.

#### 4.339.4.12 notify\_find\_search\_start()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_find_search_start () [inline], [protected]
```

Notifies a find search started.

Definition at line 57 of file hash\_policy.hpp.

#### 4.339.4.13 notify\_insert\_search\_collision()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

Definition at line 81 of file hash\_policy.hpp.

#### 4.339.4.14 notify\_insert\_search\_end()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

Definition at line 87 of file hash\_policy.hpp.



#### 4.339.4.15 notify\_insert\_search\_start()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_insert_search_start () [inline], [protected]
```

Notifies an insert search started.

Definition at line 75 of file hash\_policy.hpp.

#### 4.339.4.16 notify\_inserted()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_inserted (
 size_type num_entries) [inline], [protected]
```

Notifies an element was inserted.

Definition at line 111 of file hash\_policy.hpp.

#### 4.339.4.17 notify\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 141 of file hash\_policy.hpp.

#### 4.339.4.18 set\_load()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↵
::set_load (
 float load)
```

Sets the load; does not resize the container.

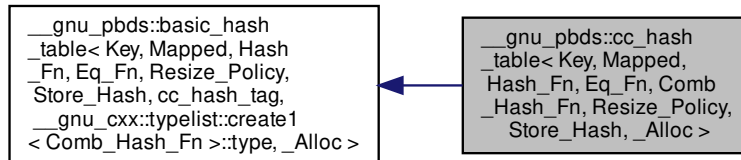
Definition at line 207 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.340 `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef Comb\_Hash\_Fn **comb\_hash\_fn**
- typedef `cc_hash_tag` **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Resize\_Policy **resize\_policy**

#### Public Member Functions

- `cc_hash_table` ()
- `cc_hash_table` (const hash\_fn &h)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- `cc_hash_table` (const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- template<typename It >  
  `cc_hash_table` (It first, It last)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch)
- template<typename It >  
  `cc_hash_table` (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_hash\_fn &ch, const resize\_policy &rp)
- **cc\_hash\_table** (const `cc_hash_table` &other)
- `cc_hash_table` & **operator=** (const `cc_hash_table` &other)
- void **swap** (`cc_hash_table` &other)

## 4.340.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy =
typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename
_Alloc = std::allocator<char>>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A collision-chaining hash-based associative container.

## Template Parameters

|                      |                                                                                                                                                                                                                      |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                            |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                            |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                                     |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                                       |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If <i>Hash_Fn</i> is not <code>null_type</code> , then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                                        |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If <i>Hash_Fn</i> is <code>null_type</code> , then the container will not compile if this value is true                                         |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                      |

Base tag choices are: `cc_hash_tag`.

Base is `basic_hash_table`.

Definition at line 204 of file `assoc_container.hpp`.

## 4.340.2 Constructor &amp; Destructor Documentation

4.340.2.1 `cc_hash_table()` [1/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn =
typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy =
typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc =
std::allocator<char>>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table () [inline]
```

Default constructor.

Definition at line 217 of file `assoc_container.hpp`.

**4.340.2.2 cc\_hash\_table()** [2/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `Hash_Fn` object of the container object.

Definition at line 221 of file `assoc_container.hpp`.

**4.340.2.3 cc\_hash\_table()** [3/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 228 of file `assoc_container.hpp`.

**4.340.2.4 cc\_hash\_table()** [4/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 236 of file `assoc_container.hpp`.

4.340.2.5 `cc_hash_table()` [5/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch,
 const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

Definition at line 245 of file `assoc_container.hpp`.

4.340.2.6 `cc_hash_table()` [6/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 253 of file `assoc_container.hpp`.

4.340.2.7 `cc_hash_table()` [7/10]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
```

```

 It last,
 const hash_fn & h) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 260 of file `assoc_container.hpp`.

#### 4.340.2.8 `cc_hash_table()` [8/10]

```

template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 271 of file `assoc_container.hpp`.

#### 4.340.2.9 `cc_hash_table()` [9/10]

```

template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↵
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↵
::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch) [inline]

```

Constructor taking `__iterators` to a range of `value_types` and some policy objects The `value_types` between `first_↵` it and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_hash_fn` will be copied by the `comb_hash_fn` object of the container object.

Definition at line 283 of file `assoc_container.hpp`.

## 4.340.2.10 cc\_hash\_table() [10/10]

```

template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<Key>::type,
typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type,
bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_hash_fn & ch,
 const resize_policy & rp) [inline]

```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_hash\_fn will be copied by the hash\_fn object of the container object, r\_eq\_fn will be copied by the eq\_fn object of the container object, r\_comb\_hash\_fn will be copied by the comb\_hash\_fn object of the container object, and r\_resize\_policy will be copied by the resize\_policy object of the container object.

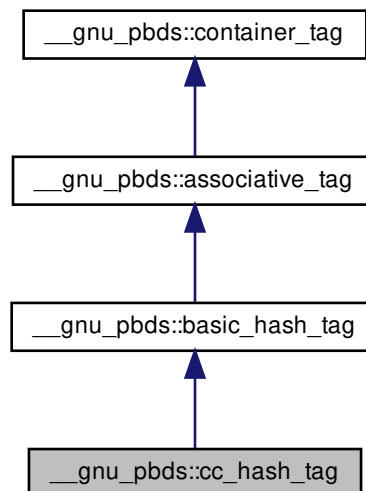
Definition at line 297 of file assoc\_container.hpp.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

## 4.341 \_\_gnu\_pbds::cc\_hash\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::cc\_hash\_tag:



#### 4.341.1 Detailed Description

Collision-chaining hash.

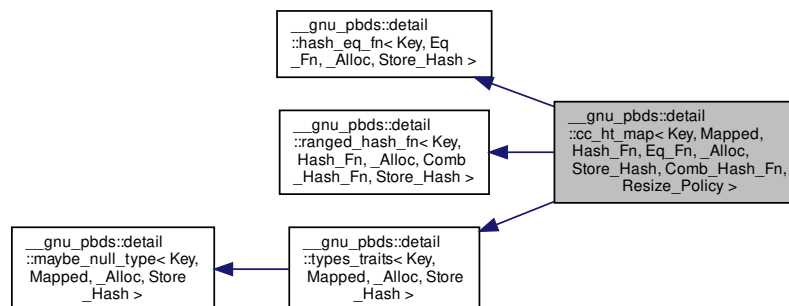
Definition at line 141 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.342 `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`:



#### Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Hash_Fn` **comb\_hash\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**



- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef point\_const\_iterator\_ **point\_const\_iterator**
- typedef point\_iterator\_ **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef Resize\_Policy **resize\_policy**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef stored\_data< value\_type, size\_type, Store\_Hash > **stored\_data\_type**
- typedef traits\_base::value\_type **value\_type**

#### Public Member Functions

- **cc\_ht\_map** (const Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &)
- **cc\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Hash\_Fn &, const Resize\_Policy &)
- **cc\_ht\_map** (const `cc_ht_map`< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It >  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Hash\_Fn & **get\_comb\_hash\_fn** ()
- const Comb\_Hash\_Fn & **get\_comb\_hash\_fn** () const
- Eq\_Fn & **get\_eq\_fn** ()
- const Eq\_Fn & **get\_eq\_fn** () const
- Hash\_Fn & **get\_hash\_fn** ()
- const Hash\_Fn & **get\_hash\_fn** () const
- Resize\_Policy & **get\_resize\_policy** ()
- const Resize\_Policy & **get\_resize\_policy** () const
- void **initialize** ()
- `std::pair`< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** (`cc_ht_map`< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- class
- class **const\_iterator\_**  
**iterator\_**

## 4.342.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename
Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
```

A collision-chaining hash-based container.

## Template Parameters

|                      |                                                                                                                                                                                                                                                         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                               |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                               |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                              |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                            |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                         |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                   |
| <i>Comb_Hash_Fn</i>  | Combining hash functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default <code>direct_mask_range_hashing</code> . |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                  |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_hash_fn`, `detail::types_traits`. (Optional: `detail::debug_↔map_base`.)

Definition at line 140 of file `cc_ht_map.hpp`.

## 4.342.2 Member Enumeration Documentation

## 4.342.2.1 anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

Definition at line 203 of file `cc_ht_map.hpp`.

#### 4.342.3 Member Function Documentation

##### 4.342.3.1 `empty()`

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::empty () const [inline]
```

True if `size() == 0`.

Definition at line 55 of file `cc_ht_map.hpp`.

##### 4.342.3.2 `get_comb_hash_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()
```

Return current `comb_hash_fn`.

Definition at line 72 of file `cc_ht_map.hpp`.

##### 4.342.3.3 `get_comb_hash_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const
```

Return current `const comb_hash_fn`.

Definition at line 78 of file `cc_ht_map.hpp`.

##### 4.342.3.4 `get_eq_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::get_eq_fn ()
```

Return current `eq_fn`.

Definition at line 60 of file `cc_ht_map.hpp`.

#### 4.342.3.5 `get_eq_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq\_fn.

Definition at line 66 of file cc\_ht\_map.hpp.

#### 4.342.3.6 `get_hash_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Hash_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash\_fn.

Definition at line 48 of file cc\_ht\_map.hpp.

#### 4.342.3.7 `get_hash_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash\_fn.

Definition at line 54 of file cc\_ht\_map.hpp.

#### 4.342.3.8 `get_resize_policy()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize\_policy.

Definition at line 84 of file cc\_ht\_map.hpp.

## 4.342.3.9 get\_resize\_policy() [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Hash_Fn , typename Resize_Policy >
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const
```

Return current const resize\_policy.

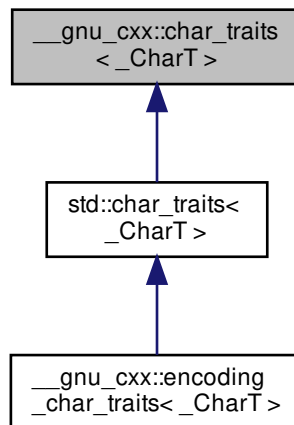
Definition at line 90 of file cc\_ht\_map.hpp.

The documentation for this class was generated from the following file:

- [cc\\_ht\\_map.hpp](#)

## 4.343 \_\_gnu\_cxx::char\_traits&lt;\_CharT&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_cxx::char\_traits<\_CharT>:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

## Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

### 4.343.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::char_traits< _CharT >
```

Base class used to implement std::char\_traits.

#### Note

For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the int\_type and state\_type typedefs, and the eof() member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace std may not be specialized for fundamental types, but classes in namespace \_\_gnu\_cxx may be.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod\_char\_traits.h.

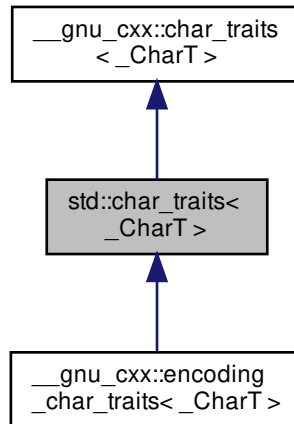
Definition at line 90 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

## 4.344 std::char\_traits&lt;\_CharT&gt; Struct Template Reference

Inheritance diagram for std::char\_traits<\_CharT>:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Char_types<_CharT>::int_type` **int\_type**
- typedef `_Char_types<_CharT>::off_type` **off\_type**
- typedef `_Char_types<_CharT>::pos_type` **pos\_type**
- typedef `_Char_types<_CharT>::state_type` **state\_type**

## Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \*\_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 4.344.1 Detailed Description

```
template<class _CharT>
struct std::char_traits< _CharT >
```

Basis for explicit traits specializations.

##### Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around `__gnu_cxx::char_traits`, it is possible to achieve a more appropriate definition by specializing `__gnu_cxx::char_traits`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character\\_types](https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types) for advice on how to make use of this class for *unusual* character types. Also, check out `include/ext/pod_char_traits.h`.

Definition at line 310 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

#### 4.345 `std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >` Struct Template Reference

##### Public Types

- typedef `__gnu_cxx::character< _Value, _Int, _St >` **char\_type**
- typedef `char_type::int_type` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `fpos< state_type >` **pos\_type**
- typedef `char_type::state_type` **state\_type**

##### Static Public Member Functions

- static void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **assign** (`char_type` \*\_\_s, size\_t \_\_n, `char_type` \_\_a)
- static int **compare** (const `char_type` \*\_\_s1, const `char_type` \*\_\_s2, size\_t \_\_n)
- static `char_type` \* **copy** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, size\_t \_\_n)
- static int\_type **eof** ()
- static bool **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static const `char_type` \* **find** (const `char_type` \*\_\_s, size\_t \_\_n, const `char_type` &\_\_a)
- static size\_t **length** (const `char_type` \*\_\_s)
- static bool **lt** (const `char_type` &\_\_c1, const `char_type` &\_\_c2)
- static `char_type` \* **move** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, size\_t \_\_n)
- static int\_type **not\_eof** (const int\_type &\_\_c)
- static `char_type` **to\_char\_type** (const int\_type &\_\_i)
- static int\_type **to\_int\_type** (const `char_type` &\_\_c)



## 4.345.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St>
struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
```

char\_traits<\_\_gnu\_cxx::character> specialization.

Definition at line 97 of file pod\_char\_traits.h.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 4.346 std::char\_traits&lt; char &gt; Struct Template Reference

## Public Types

- typedef char **char\_type**
- typedef int **int\_type**
- typedef [streamoff](#) **off\_type**
- typedef [streampos](#) **pos\_type**
- typedef mbstate\_t **state\_type**

## Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **assign** (char\_type \*\_\_s, size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **eof** () noexcept
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2) noexcept
- static constexpr const char\_type \* **find** (const char\_type \*\_\_s, size\_t \_\_n, const char\_type &\_\_a)
- static constexpr size\_t **length** (const char\_type \*\_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2) noexcept
- static constexpr char\_type \* **move** (char\_type \*\_\_s1, const char\_type \*\_\_s2, size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c) noexcept
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c) noexcept
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c) noexcept

## 4.346.1 Detailed Description

```
template<>
struct std::char_traits< char >
```

## 21.1.3.1 char\_traits specializations

Definition at line 316 of file char\_traits.h.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

#### 4.347 `std::char_traits< wchar_t >` Struct Template Reference

##### Public Types

- typedef `wchar_t` **char\_type**
- typedef `wint_t` **int\_type**
- typedef `streamoff` **off\_type**
- typedef `wstreampos` **pos\_type**
- typedef `mbstate_t` **state\_type**

##### Static Public Member Functions

- static constexpr void **assign** (`char_type` &\_\_c1, const `char_type` &\_\_c2) noexcept
- static constexpr `char_type` \* **assign** (`char_type` \*\_\_s, `size_t` \_\_n, `char_type` \_\_a)
- static constexpr int **compare** (const `char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static constexpr `char_type` \* **copy** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static constexpr `int_type` **eof** () noexcept
- static constexpr bool **eq** (const `char_type` &\_\_c1, const `char_type` &\_\_c2) noexcept
- static constexpr bool **eq\_int\_type** (const `int_type` &\_\_c1, const `int_type` &\_\_c2) noexcept
- static constexpr const `char_type` \* **find** (const `char_type` \*\_\_s, `size_t` \_\_n, const `char_type` &\_\_a)
- static constexpr `size_t` **length** (const `char_type` \*\_\_s)
- static constexpr bool **lt** (const `char_type` &\_\_c1, const `char_type` &\_\_c2) noexcept
- static constexpr `char_type` \* **move** (`char_type` \*\_\_s1, const `char_type` \*\_\_s2, `size_t` \_\_n)
- static constexpr `int_type` **not\_eof** (const `int_type` &\_\_c) noexcept
- static constexpr `char_type` **to\_char\_type** (const `int_type` &\_\_c) noexcept
- static constexpr `int_type` **to\_int\_type** (const `char_type` &\_\_c) noexcept

##### 4.347.1 Detailed Description

template<>

struct `std::char_traits< wchar_t >`

##### 21.1.3.2 `char_traits` specializations

Definition at line 451 of file `char_traits.h`.

The documentation for this struct was generated from the following file:

- [char\\_traits.h](#)

#### 4.348 `__gnu_cxx::character< _Value, _Int, _St >` Struct Template Reference

##### Public Types

- typedef `character< _Value, _Int, _St >` **char\_type**
- typedef `_Int` **int\_type**
- typedef `_St` **state\_type**
- typedef `_Value` **value\_type**

### Static Public Member Functions

- `template<typename V2 >`  
`static char\_type from` (`const V2 &v`)
- `template<typename V2 >`  
`static V2 to` (`const char\_type &c`)

### Public Attributes

- `value_type` **value**

#### 4.348.1 Detailed Description

```
template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >
```

A POD class that serves as a character abstraction class.

Definition at line 49 of file `pod_char_traits.h`.

The documentation for this struct was generated from the following file:

- [pod\\_char\\_traits.h](#)

## 4.349 `std::chi_squared_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- `typedef _RealType` [result\\_type](#)

## Public Member Functions

- **chi\_squared\_distribution** (\_RealType \_\_n)
- **chi\_squared\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- \_RealType n () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) param () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)
- bool [operator==](#) (const [chi\\_squared\\_distribution](#) &\_\_d1, const [chi\\_squared\\_distribution](#) &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::chi\\_squared\\_distribution](#)< \_RealType1 > &\_\_x)

## 4.349.1 Detailed Description

```
template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >
```

A [chi\\_squared\\_distribution](#) random number distribution.

The formula for the normal probability mass function is  $p(x|n) = \frac{x^{(n/2)-1} e^{-x/2}}{\Gamma(n/2) 2^{n/2}}$

Definition at line 2630 of file random.h.

## 4.349.2 Member Typedef Documentation

#### 4.349.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2633 of file random.h.

### 4.349.3 Member Function Documentation

#### 4.349.3.1 max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2724 of file random.h.

References std::numeric\_limits<\_Tp>::max().

#### 4.349.3.2 min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2717 of file random.h.

#### 4.349.3.3 n()

```
template<typename _RealType = double>
_RealType std::chi_squared_distribution< _RealType >::n () const [inline]
```

Definition at line 2690 of file random.h.

#### 4.349.3.4 operator()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::chi_squared_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 2732 of file random.h.

#### 4.349.3.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2697 of file random.h.

#### 4.349.3.6 param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2705 of file random.h.

References `std::gamma_distribution< _RealType >::param()`.

#### 4.349.3.7 reset()

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 2683 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

## 4.349.4 Friends And Related Function Documentation

## 4.349.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::chi_squared_distribution<_RealType1> & __x) [friend]
```

Inserts a chi\_squared\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

|      |                                                        |
|------|--------------------------------------------------------|
| __os | An output stream.                                      |
| __x  | A chi_squared_distribution random number distribution. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 4.349.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
 const chi_squared_distribution<_RealType> & __d1,
 const chi_squared_distribution<_RealType> & __d2) [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2783 of file random.h.

## 4.349.4.3 operator&gt;&gt;

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::chi_squared_distribution<_RealType1> & __x) [friend]
```

Extracts a chi\_squared\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                         |                                                                         |
|-------------------------|-------------------------------------------------------------------------|
| <code>_↵<br/>_is</code> | An input stream.                                                        |
| <code>_↵<br/>_x</code>  | A <code>chi_squared_distribution</code> random number generator engine. |

## Returns

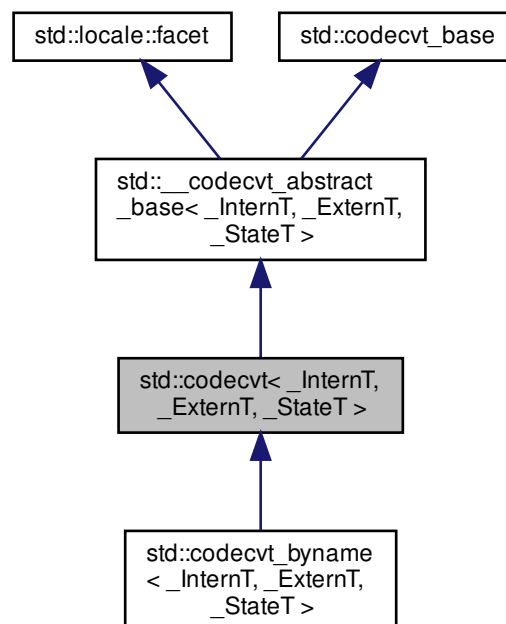
The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.350 `std::codecvt<_InternT, _ExternT, _StateT >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, _StateT >`:





## Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

## Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale` `_M_c_locale_codecvt`

### 4.350.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt< _InternT, _ExternT, _StateT >
```

Primary class template codecvt.

NB: Generic, mostly useless implementation.

Definition at line 274 of file codecvt.h.

### 4.350.2 Member Function Documentation

#### 4.350.2.1 `do_out()`

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

#### See also

`out` for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< `_InternT`, `_ExternT`, `_StateT` >.

## 4.350.2.2 in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

#### 4.350.2.3 out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

##### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

##### Returns

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

## 4.350.2.4 unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

## Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

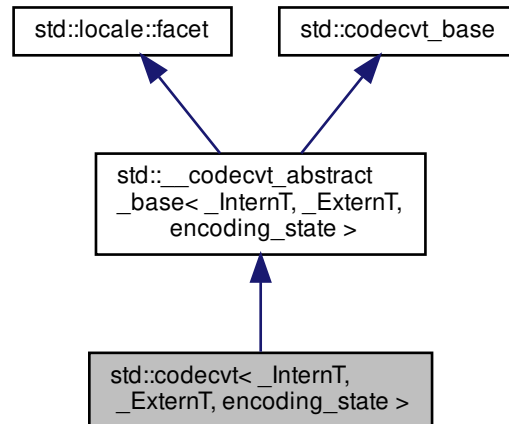
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 4.351 `std::codecvt<_InternT, _ExternT, encoding_state >` Class Template Reference

Inheritance diagram for `std::codecvt<_InternT, _ExternT, encoding_state >`:



#### Public Types

- typedef `state_type::descriptor_type` **descriptor\_type**
- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `__gnu_cxx::encoding_state` **state\_type**

#### Public Member Functions

- **codecvt** (`size_t __refs=0`)
- **codecvt** (`state_type &__enc, size_t __refs=0`)
- **bool always\_noconv** () const throw ()
- **int encoding** () const throw ()
- **result in** (`state_type &__state, const extern_type * __from, const extern_type * __from_end, const extern_type * & __from_next, intern_type * __to, intern_type * __to_end, intern_type * & __to_next`) const
- **int length** (`state_type &__state, const extern_type * __from, const extern_type * __end, size_t __max`) const
- **int max\_length** () const throw ()
- **result out** (`state_type &__state, const intern_type * __from, const intern_type * __from_end, const intern_type * & __from_next, extern_type * __to, extern_type * __to_end, extern_type * & __to_next`) const
- **result unshift** (`state_type &__state, extern_type * __to, extern_type * __to_end, extern_type * & __to_next`) const

#### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* & \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* & \_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* & \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* & \_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

## 4.351.1 Detailed Description

```
template<typename _InternT, typename _ExternT>
class std::codecvt<_InternT, _ExternT, encoding_state >
```

codecvt<InternT, \_ExternT, encoding\_state> specialization.

Definition at line 233 of file codecvt\_specializations.h.

## 4.351.2 Member Function Documentation

## 4.351.2.1 do\_out()

```
template<typename _InternT , typename _ExternT >
codecvt_base::result std::codecvt<_InternT, _ExternT, encoding_state >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type * & __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type * & __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, encoding\\_state>](#).

Definition at line 309 of file `codecvt_specializations.h`.

#### 4.351.2.2 in()

```
result std::__codecvt_abstract_base<_InternT, _ExternT, encoding_state>::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

##### Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |



**Returns**

codecvt\_base::result.

Definition at line 196 of file codecvt.h.

**4.351.2.3 out()**

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <i>__state</i>     | Persistent conversion state data.    |
| <i>__from</i>      | Start of input.                      |
| <i>__from_end</i>  | End of input.                        |
| <i>__from_next</i> | Returns start of unconverted data.   |
| <i>__to</i>        | Start of output buffer.              |
| <i>__to_end</i>    | End of output buffer.                |
| <i>__to_next</i>   | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

#### 4.351.2.4 unshift()

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

##### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

##### Returns

`codecvt_base::result`.

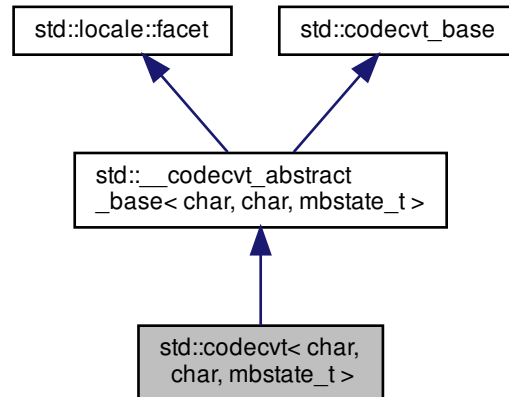
Definition at line 155 of file `codecvt.h`.

The documentation for this class was generated from the following file:

- [codecvt\\_specializations.h](#)

## 4.352 std::codecvt&lt; char, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for std::codecvt< char, char, mbstate\_t >:



## Public Types

- typedef char **extern\_type**
- typedef char **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_to\_next) const

## Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

### Friends

- class **messages**< char >

#### 4.352.1 Detailed Description

```
template<>
class std::codecvt< char, char, mbstate_t >
```

class codecvt<char, char, mbstate\_t> specialization.

Definition at line 338 of file codecvt.h.

#### 4.352.2 Member Function Documentation

4.352.2.1 `do_out()`

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

**See also**

`out` for more information.

Implements `std::__codecvt_abstract_base< char, char, mbstate_t >`.

4.352.2.2 `in()`

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

**4.352.2.3 out()**

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

**Parameters**

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |
| <code>__to</code>        | Start of output buffer.            |

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**4.352.2.4 unshift()**

```
result std::__codecvt_abstract_base< char , char , mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

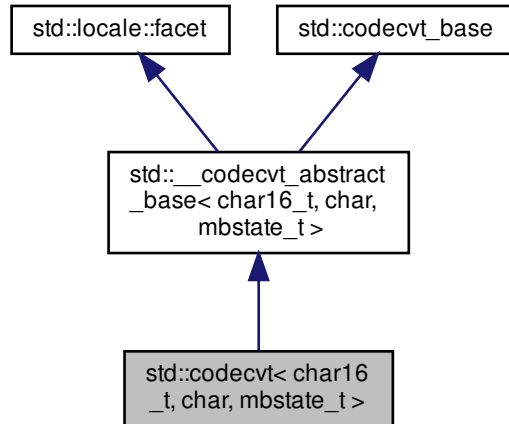
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 4.353 std::codecvt< char16\_t, char, mbstate\_t > Class Template Reference

Inheritance diagram for std::codecvt< char16\_t, char, mbstate\_t >:



#### Public Types

- typedef char **extern\_type**
- typedef char16\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

#### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

#### Static Public Attributes

- static locale::id **id**



## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.353.1 Detailed Description

```
template<>
class std::codecvt< char16_t, char, mbstate_t >
```

Class `codecvt<char16_t, char, mbstate_t>` specialization.

Converts between UTF-16 and UTF-8.

Definition at line 467 of file `codecvt.h`.

## 4.353.2 Member Function Documentation

#### 4.353.2.1 do\_out()

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

##### See also

[out](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< char16\_t, char, mbstate\_t >.

#### 4.353.2.2 in()

```
result std::__codecvt_abstract_base< char16_t , char , mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling [codecvt::do\\_in](#).

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of [codecvt\\_base::result](#). If all the input is converted, returns [codecvt\\_base::ok](#). If no conversion is necessary, returns [codecvt\\_base::noconv](#). If the input ends early or there is insufficient space in the output, returns [codecvt\\_base::partial](#). Otherwise the conversion failed and [codecvt\\_base::error](#) is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.353.2.3 `out()`

```
result std::__codecvt_abstract_base< char16_t , char , mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |
| <code>__to</code>        | Start of output buffer.            |

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**4.353.2.4 unshift()**

```
result std::__codecvt_abstract_base< char16_t , char , mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

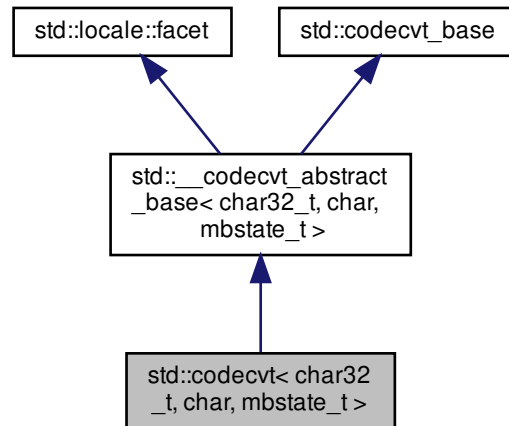
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.354 std::codecvt&lt; char32\_t, char, mbstate\_t &gt; Class Template Reference

Inheritance diagram for std::codecvt< char32\_t, char, mbstate\_t >:



## Public Types

- typedef char **extern\_type**
- typedef char32\_t **intern\_type**
- typedef codecvt\_base::result **result**
- typedef mbstate\_t **state\_type**

## Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Public Attributes

- static locale::id **id**

### Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### 4.354.1 Detailed Description

```
template<>
class std::codecvt< char32_t, char, mbstate_t >
```

Class codecvt<char32\_t, char, mbstate\_t> specialization.

Converts between UTF-32 and UTF-8.

Definition at line 524 of file codecvt.h.

#### 4.354.2 Member Function Documentation

4.354.2.1 `do_out()`

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

## See also

`out` for more information.

Implements `std::__codecvt_abstract_base< char32_t, char, mbstate_t >`.

4.354.2.2 `in()`

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

4.354.2.3 `out()`

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The `state` argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how `state` is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |
| <code>__to</code>        | Start of output buffer.            |



**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**4.354.2.4 unshift()**

```
result std::__codecvt_abstract_base< char32_t , char , mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

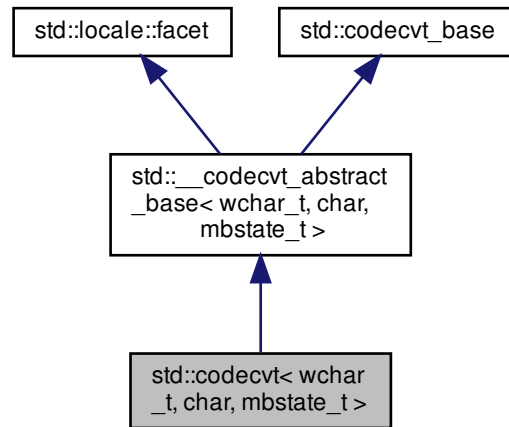
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

### 4.355 `std::codecvt< wchar_t, char, mbstate_t >` Class Template Reference

Inheritance diagram for `std::codecvt< wchar_t, char, mbstate_t >`:



#### Public Types

- typedef char **extern\_type**
- typedef wchar\_t **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef mbstate\_t **state\_type**

#### Public Member Functions

- **codecvt** (size\_t \_\_refs=0)
- **codecvt** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- int **length** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- result **unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

#### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## Friends

- class **messages< wchar\_t >**

## 4.355.1 Detailed Description

template<>

class `std::codecvt< wchar_t, char, mbstate_t >`

Class `codecvt<wchar_t, char, mbstate_t>` specialization.

Converts between narrow and wide characters in the native character set

Definition at line 401 of file `codecvt.h`.

## 4.355.2 Member Function Documentation

#### 4.355.2.1 do\_out()

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This function is a hook for derived classes to change the value returned.

##### See also

out for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base< wchar\\_t, char, mbstate\\_t >](#).

#### 4.355.2.2 in()

```
result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern\_type to output string of intern\_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do\_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

`codecvt_base::result`.

Definition at line 196 of file `codecvt.h`.

## 4.355.2.3 out()

```
result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This is analogous to `wcsrtombs`. It does this by calling `codecvt::do_out`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                    |
|--------------------------|------------------------------------|
| <code>__state</code>     | Persistent conversion state data.  |
| <code>__from</code>      | Start of input.                    |
| <code>__from_end</code>  | End of input.                      |
| <code>__from_next</code> | Returns start of unconverted data. |
| <code>__to</code>        | Start of output buffer.            |

**Returns**

codecvt\_base::result.

Definition at line 116 of file codecvt.h.

**4.355.2.4 unshift()**

```
result std::__codecvt_abstract_base< wchar_t , char , mbstate_t >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do\_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt\_base::result. If the state could be reset and data written, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the output has insufficient space, returns codecvt\_base::partial. Otherwise the reset failed and codecvt\_base::error is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

**Returns**

codecvt\_base::result.

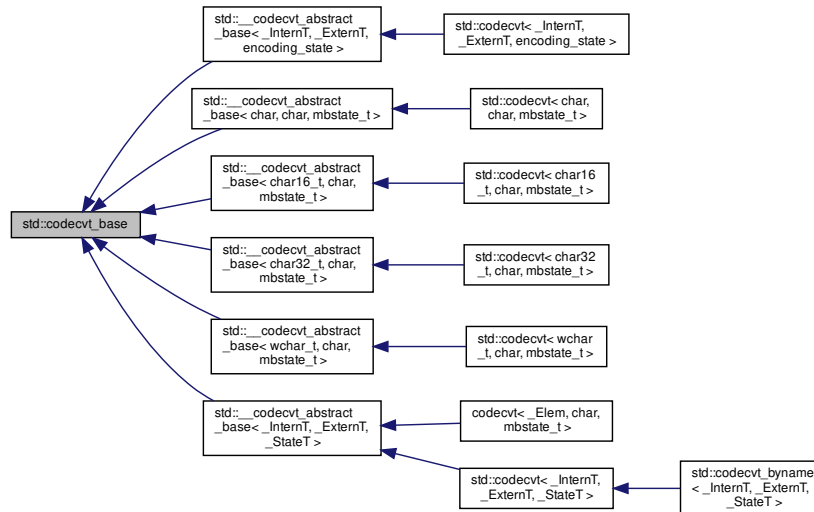
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.356 std::codecvt\_base Class Reference

Inheritance diagram for std::codecvt\_base:



## Public Types

- enum **result** { **ok**, **partial**, **error**, **noconv** }

## 4.356.1 Detailed Description

Empty base class for codecvt facet [22.2.1.5].

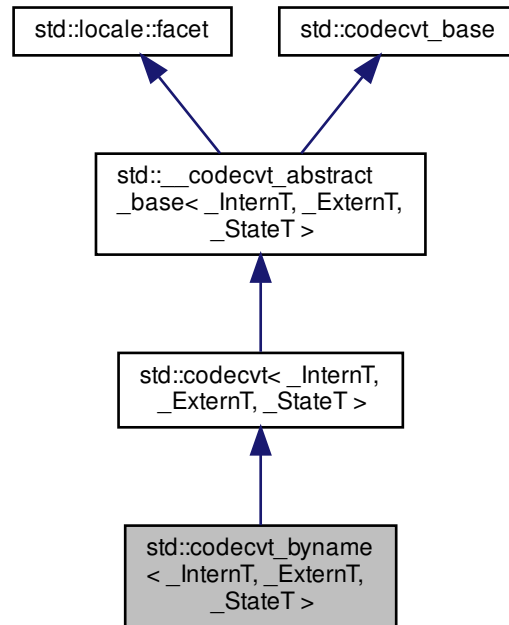
Definition at line 46 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

#### 4.357 std::codecvt\_byname< \_InternT, \_ExternT, \_StateT > Class Template Reference

Inheritance diagram for std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >:



##### Public Types

- typedef `_ExternT` **extern\_type**
- typedef `_InternT` **intern\_type**
- typedef `codecvt_base::result` **result**
- typedef `_StateT` **state\_type**

##### Public Member Functions

- **codecvt\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **codecvt\_byname** (const [string](#) & \_\_s, size\_t \_\_refs=0)
- bool **always\_noconv** () const throw ()
- int **encoding** () const throw ()
- result **in** (state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_from\_end, const extern\_type \* \_\_from\_next, intern\_type \* \_\_to, intern\_type \* \_\_to\_end, intern\_type \* \_\_to\_next) const
- int **length** (state\_type & \_\_state, const extern\_type \* \_\_from, const extern\_type \* \_\_end, size\_t \_\_max) const
- int **max\_length** () const throw ()
- result **out** (state\_type & \_\_state, const intern\_type \* \_\_from, const intern\_type \* \_\_from\_end, const intern\_type \* \_\_from\_next, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_to\_next) const
- result **unshift** (state\_type & \_\_state, extern\_type \* \_\_to, extern\_type \* \_\_to\_end, extern\_type \* \_\_to\_next) const



## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual bool **do\_always\_noconv** () const throw ()
- virtual int **do\_encoding** () const throw ()
- virtual result **do\_in** (state\_type &\_\_state, const extern\_type \*\_\_from, const extern\_type \*\_\_from\_end, const extern\_type \*\_\_&\_\_from\_next, intern\_type \*\_\_to, intern\_type \*\_\_to\_end, intern\_type \*\_\_&\_\_to\_next) const
- virtual int **do\_length** (state\_type &, const extern\_type \*\_\_from, const extern\_type \*\_\_end, size\_t \_\_max) const
- virtual int **do\_max\_length** () const throw ()
- virtual result **do\_out** (state\_type &\_\_state, const intern\_type \*\_\_from, const intern\_type \*\_\_from\_end, const intern\_type \*\_\_&\_\_from\_next, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const
- virtual result **do\_unshift** (state\_type &\_\_state, extern\_type \*\_\_to, extern\_type \*\_\_to\_end, extern\_type \*\_\_&\_\_to\_next) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_codecvt**

## 4.357.1 Detailed Description

```
template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >
```

class codecvt\_byname [22.2.1.6].

Definition at line 696 of file codecvt.h.

## 4.357.2 Member Function Documentation

#### 4.357.2.1 do\_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [protected], [virtual], [inherited]
```

Convert from internal to external character set.

Converts input string of `intern_type` to output string of `extern_type`. This function is a hook for derived classes to change the value returned.

##### See also

[do\\_in](#) for more information.

Implements [std::\\_\\_codecvt\\_abstract\\_base](#)< `_InternT`, `_ExternT`, `_StateT` >.

#### 4.357.2.2 in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
 state_type & __state,
 const extern_type * __from,
 const extern_type * __from_end,
 const extern_type *& __from_next,
 intern_type * __to,
 intern_type * __to_end,
 intern_type *& __to_next) const [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of `extern_type` to output string of `intern_type`. This is analogous to `mbsrtowcs`. It does this by calling `codecvt::do_in`.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in `[from,from_end)` are converted and written to `[to,to_end)`. `from_next` and `to_next` are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, `from_next` and `to_next` are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of `codecvt_base::result`. If all the input is converted, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the input ends early or there is insufficient space in the output, returns `codecvt_base::partial`. Otherwise the conversion failed and `codecvt_base::error` is returned.

## Parameters

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

## Returns

codecvt\_base::result.

Definition at line 196 of file codecvt.h.

## 4.357.2.3 out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
 state_type & __state,
 const intern_type * __from,
 const intern_type * __from_end,
 const intern_type *& __from_next,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern\_type to output string of extern\_type. This is analogous to wcsrtombs. It does this by calling codecvt::do\_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from\_end) are converted and written to [to,to\_end). from\_next and to\_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from\_next and to\_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt\_base::result. If all the input is converted, returns codecvt\_base::ok. If no conversion is necessary, returns codecvt\_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt\_base::partial. Otherwise the conversion failed and codecvt\_base::error is returned.

**Parameters**

|                          |                                      |
|--------------------------|--------------------------------------|
| <code>__state</code>     | Persistent conversion state data.    |
| <code>__from</code>      | Start of input.                      |
| <code>__from_end</code>  | End of input.                        |
| <code>__from_next</code> | Returns start of unconverted data.   |
| <code>__to</code>        | Start of output buffer.              |
| <code>__to_end</code>    | End of output buffer.                |
| <code>__to_next</code>   | Returns start of unused output area. |

**Returns**

`codecvt_base::result`.

Definition at line 116 of file `codecvt.h`.

**4.357.2.4 unshift()**

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
 state_type & __state,
 extern_type * __to,
 extern_type * __to_end,
 extern_type *& __to_next) const [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling `codecvt::do_unshift()`.

For example, if 4 external characters always converted to 1 internal character, and input to `in()` had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of `codecvt_base::result`. If the state could be reset and data written, returns `codecvt_base::ok`. If no conversion is necessary, returns `codecvt_base::noconv`. If the output has insufficient space, returns `codecvt_base::partial`. Otherwise the reset failed and `codecvt_base::error` is returned.

**Parameters**

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__state</code>   | Persistent conversion state data.    |
| <code>__to</code>      | Start of output buffer.              |
| <code>__to_end</code>  | End of output buffer.                |
| <code>__to_next</code> | Returns start of unused output area. |

## Returns

codecvt\_base::result.

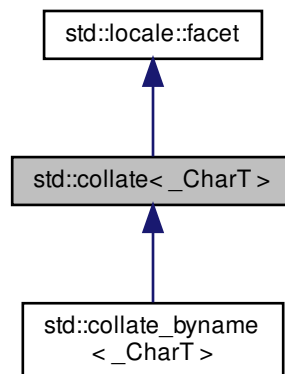
Definition at line 155 of file codecvt.h.

The documentation for this class was generated from the following file:

- [codecvt.h](#)

## 4.358 std::collate&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::collate<\_CharT>:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef [basic\\_string](#)<\_CharT> [string\\_type](#)

### Public Member Functions

- [collate](#) (size\_t \_\_refs=0)
- [collate](#) (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- int **\_M\_compare** (const \_CharT \*, const \_CharT \*) const throw ()
- template<>  
int **\_M\_compare** (const char \*, const char \*) const throw()
- template<>  
int **\_M\_compare** (const wchar\_t \*, const wchar\_t \*) const throw()
- size\_t **\_M\_transform** (\_CharT \*, const \_CharT \*, size\_t) const throw ()
- template<>  
size\_t **\_M\_transform** (char \*, const char \*, size\_t) const throw()
- template<>  
size\_t **\_M\_transform** (wchar\_t \*, const wchar\_t \*, size\_t) const throw()
- int [compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- long [hash](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- [string\\_type transform](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

### Static Public Attributes

- static [locale::id](#) id

### Protected Member Functions

- virtual [~collate](#) ()
- virtual int [do\\_compare](#) (const \_CharT \* \_\_lo1, const \_CharT \* \_\_hi1, const \_CharT \* \_\_lo2, const \_CharT \* \_\_hi2) const
- virtual long [do\\_hash](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const
- virtual [string\\_type do\\_transform](#) (const \_CharT \* \_\_lo, const \_CharT \* \_\_hi) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_collate**

#### 4.358.1 Detailed Description

```
template<typename _CharT>
class std::collate<_CharT>
```

Facet for localized string comparison.

This facet encapsulates the code to compare strings in a localized manner.

The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

Definition at line 644 of file locale\_classes.h.

#### 4.358.2 Member Typedef Documentation

##### 4.358.2.1 char\_type

```
template<typename _CharT>
typedef _CharT std::collate<_CharT>::char_type
```

Public typedefs.

Definition at line 650 of file locale\_classes.h.

##### 4.358.2.2 string\_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::collate<_CharT>::string_type
```

Public typedefs.

Definition at line 651 of file locale\_classes.h.

#### 4.358.3 Constructor & Destructor Documentation

##### 4.358.3.1 collate() [1/2]

```
template<typename _CharT>
std::collate<_CharT>::collate (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 671 of file locale\_classes.h.

**4.358.3.2 collate()** [2/2]

```
template<typename _CharT>
std::collate< _CharT >::collate (
 __c_locale __cloc,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__refs</code> | Passed to the base facet class. |

Definition at line 685 of file locale\_classes.h.

**4.358.3.3 ~collate()**

```
template<typename _CharT>
virtual std::collate< _CharT >::~~collate () [inline], [protected], [virtual]
```

Destructor.

Definition at line 748 of file locale\_classes.h.

**4.358.4 Member Function Documentation****4.358.4.1 compare()**

```
template<typename _CharT>
int std::collate< _CharT >::compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [inline]
```

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.



## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 702 of file `locale_classes.h`.

## 4.358.4.2 do\_compare()

```
template<typename _CharT>
int std::collate<_CharT>::do_compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [protected], [virtual]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

## See also

`compare()`.

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 161 of file `locale_classes.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

#### 4.358.4.3 do\_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

##### Parameters

|                     |                  |
|---------------------|------------------|
| $\leftarrow$<br>_lo | Start of string. |
| $\leftarrow$<br>_hi | End of string.   |

##### Returns

Hash value.

Definition at line 256 of file locale\_classes.tcc.

#### 4.358.4.4 do\_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

##### Parameters

|                     |        |
|---------------------|--------|
| $\leftarrow$<br>_lo | Start. |
| $\leftarrow$<br>_hi | End.   |

##### Returns

transformed string.

Definition at line 200 of file locale\_classes.tcc.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::basic_string<_CharT, _Traits, _Alloc>::data()`, and `std::basic_string<_CharT, _Traits, _Alloc>::length()`.

#### 4.358.4.5 hash()

```
template<typename _CharT>
long std::collate<_CharT>::hash (
 const _CharT * __lo,
 const _CharT * __hi) const [inline]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

##### Parameters

|                        |                  |
|------------------------|------------------|
| <code>↔<br/>_lo</code> | Start of string. |
| <code>↔<br/>_hi</code> | End of string.   |

##### Returns

Hash value.

Definition at line 735 of file locale\_classes.h.

#### 4.358.4.6 transform()

```
template<typename _CharT>
string_type std::collate<_CharT>::transform (
 const _CharT * __lo,
 const _CharT * __hi) const [inline]
```

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

## Parameters

|                         |                  |
|-------------------------|------------------|
| <code>_↵<br/>_lo</code> | Start of string. |
| <code>_↵<br/>_hi</code> | End of string.   |

## Returns

Transformed string\_type.

Definition at line 721 of file locale\_classes.h.

## 4.358.5 Member Data Documentation

## 4.358.5.1 id

```
template<typename _CharT>
locale::id std::collate< _CharT >::id [static]
```

Numpunct facet id.

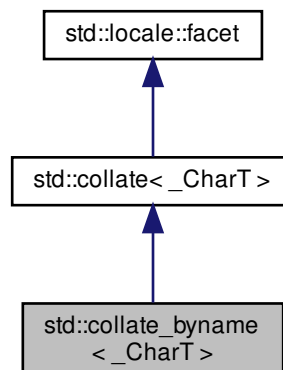
Definition at line 661 of file locale\_classes.h.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 4.359 std::collate\_byname&lt; \_CharT &gt; Class Template Reference

Inheritance diagram for std::collate\_byname< \_CharT >:



## Public Types

- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string](#)< `_CharT` > [string\\_type](#)

## Public Member Functions

- **collate\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **collate\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- **int \_M\_compare** (const `_CharT` \*, const `_CharT` \*) const throw ()
- `template<>`  
**int \_M\_compare** (const char \*, const char \*) const throw()
- `template<>`  
**int \_M\_compare** (const `wchar_t` \*, const `wchar_t` \*) const throw()
- **size\_t \_M\_transform** (`_CharT` \*, const `_CharT` \*, size\_t) const throw ()
- `template<>`  
**size\_t \_M\_transform** (char \*, const char \*, size\_t) const throw()
- `template<>`  
**size\_t \_M\_transform** (`wchar_t` \*, const `wchar_t` \*, size\_t) const throw()
- **int compare** (const `_CharT` \*\_\_lo1, const `_CharT` \*\_\_hi1, const `_CharT` \*\_\_lo2, const `_CharT` \*\_\_hi2) const
- **long hash** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const
- **string\_type transform** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual **int do\_compare** (const `_CharT` \*\_\_lo1, const `_CharT` \*\_\_hi1, const `_CharT` \*\_\_lo2, const `_CharT` \*\_\_hi2) const
- virtual **long do\_hash** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const
- virtual **string\_type do\_transform** (const `_CharT` \*\_\_lo, const `_CharT` \*\_\_hi) const

## Static Protected Member Functions

- static `_c_locale` **\_S\_clone\_c\_locale** (`_c_locale` &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (`_c_locale` &\_\_cloc, const char \*\_\_s, `_c_locale` \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (`_c_locale` &\_\_cloc)
- static `_c_locale` **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static `_c_locale` **\_S\_lc\_ctype\_c\_locale** (`_c_locale` \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale __M_c_locale_collate`

### 4.359.1 Detailed Description

```
template<typename _CharT>
class std::collate_byname< _CharT >
```

class `collate_byname` [22.2.4.2].

Definition at line 818 of file `locale_classes.h`.

### 4.359.2 Member Typedef Documentation

#### 4.359.2.1 `char_type`

```
template<typename _CharT >
typedef _CharT std::collate_byname< _CharT >::char_type
```

Public typedefs.

Definition at line 823 of file `locale_classes.h`.

#### 4.359.2.2 `string_type`

```
template<typename _CharT >
typedef basic_string<_CharT> std::collate_byname< _CharT >::string_type
```

Public typedefs.

Definition at line 824 of file `locale_classes.h`.

### 4.359.3 Member Function Documentation

#### 4.359.3.1 `compare()`

```
template<typename _CharT>
int std::collate< _CharT >::compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [inline], [inherited]
```

Compare two strings.

This function compares two strings and returns the result by calling `collate::do_compare()`.

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 702 of file `locale_classes.h`.

## 4.359.3.2 do\_compare()

```
template<typename _CharT>
int std::collate< _CharT >::do_compare (
 const _CharT * __lo1,
 const _CharT * __hi1,
 const _CharT * __lo2,
 const _CharT * __hi2) const [protected], [virtual], [inherited]
```

Compare two strings.

This function is a hook for derived classes to change the value returned.

## See also

`compare()`.

## Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__lo1</code> | Start of string 1. |
| <code>__hi1</code> | End of string 1.   |
| <code>__lo2</code> | Start of string 2. |
| <code>__hi2</code> | End of string 2.   |

## Returns

1 if `string1 > string2`, -1 if `string1 < string2`, else 0.

Definition at line 161 of file `locale_classes.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.



#### 4.359.3.3 do\_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual], [inherited]
```

Return hash of a string.

This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

##### Parameters

|                                                                                               |                  |
|-----------------------------------------------------------------------------------------------|------------------|
|  <i>__lo</i> | Start of string. |
|  <i>__hi</i> | End of string.   |

##### Returns

Hash value.

Definition at line 256 of file locale\_classes.tcc.



#### 4.359.3.4 do\_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
 const _CharT * __lo,
 const _CharT * __hi) const [protected], [virtual], [inherited]
```

Transform string to comparable form.

This function is a hook for derived classes to change the value returned.

##### Parameters

|                                                                                                 |        |
|-------------------------------------------------------------------------------------------------|--------|
|  <i>__lo</i> | Start. |
|  <i>__hi</i> | End.   |

##### Returns

transformed string.



Definition at line 200 of file locale\_classes.tcc.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`, `std::basic_string< _CharT, _Traits, _Alloc >::data()`, and `std::basic_string< _CharT, _Traits, _Alloc >::length()`.

#### 4.359.3.5 hash()

```
template<typename _CharT>
long std::collate< _CharT >::hash (
 const _CharT * __lo,
 const _CharT * __hi) const [inline], [inherited]
```

Return hash of a string.

This function computes and returns a hash on the input string. It does so by returning `collate::do_hash()`.

##### Parameters

|                   |                  |
|-------------------|------------------|
| <code>__lo</code> | Start of string. |
| <code>__hi</code> | End of string.   |

##### Returns

Hash value.

Definition at line 735 of file locale\_classes.h.

#### 4.359.3.6 transform()

```
template<typename _CharT>
string_type std::collate< _CharT >::transform (
 const _CharT * __lo,
 const _CharT * __hi) const [inline], [inherited]
```

Transform string to comparable form.

This function is a wrapper for `strxfrm` functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning `collate::do_transform()`.

**Parameters**

|                         |                  |
|-------------------------|------------------|
| <code>_↵<br/>_lo</code> | Start of string. |
| <code>_↵<br/>_hi</code> | End of string.   |

**Returns**

Transformed string\_type.

Definition at line 721 of file locale\_classes.h.

**4.359.4 Member Data Documentation****4.359.4.1 id**

```
template<typename _CharT>
locale::id std::collate<_CharT >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 661 of file locale\_classes.h.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

**4.360 std::common\_type<\_Tp > Struct Template Reference**

Inherited by std::common\_type<\_Tp1, \_Tp2 >.

**4.360.1 Detailed Description**

```
template<typename... _Tp>
struct std::common_type<_Tp >
```

common\_type

Definition at line 2215 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.361 `std::common_type< chrono::duration< _Rep, _Period > >` Struct Template Reference

### Public Types

- using **type** = `chrono::duration< typename common_type< _Rep >::type, typename _Period::type >`

#### 4.361.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period > >
```

Specialization of `common_type` for one `chrono::duration` type.

Definition at line 125 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

## 4.362 `std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >` Struct Template Reference

### Public Types

- using **type** = `chrono::duration< typename common_type< _Rep >::type, typename _Period::type >`

#### 4.362.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
```

Specialization of `common_type` for two identical `chrono::duration` types.

Definition at line 115 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

## 4.363 `std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >` Struct Template Reference

Inherits `__duration_common_type< common_type< _Rep1, _Rep2 >, _Period1::type, _Period2::type >`.

#### 4.363.1 Detailed Description

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
```

Specialization of `common_type` for `chrono::duration` types.

Definition at line 105 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### 4.364 `std::common_type< chrono::time_point< _Clock, _Duration > >` Struct Template Reference

##### Public Types

- using **type** = [chrono::time\\_point](#)< `_Clock`, `_Duration` >

#### 4.364.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration > >
```

Specialization of `common_type` for one `chrono::time_point` type.

Definition at line 165 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### 4.365 `std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >` Struct Template Reference

##### Public Types

- using **type** = [chrono::time\\_point](#)< `_Clock`, `_Duration` >

#### 4.365.1 Detailed Description

```
template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
```

Specialization of `common_type` for two identical `chrono::time_point` types.

Definition at line 158 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### **4.366** `std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >` Struct Template Reference

Inherits `__timepoint_common_type< common_type< _Duration1, _Duration2 >, _Clock >`.

#### 4.366.1 Detailed Description

```
template<typename _Clock, typename _Duration1, typename _Duration2>
struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
```

Specialization of `common_type` for `chrono::time_point` types.

Definition at line 150 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### **4.367** `std::complex< _Tp >` Struct Template Reference

##### Public Types

- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr `complex` (const `_Tp` &\_\_r=\_Tp(), const `_Tp` &\_\_i=\_Tp())
- constexpr `complex` (const `complex` &)=default
- template<typename `_Up` >  
constexpr `complex` (const `complex`< `_Up` > &\_\_z)
- constexpr `complex` `__rep` () const
- `_GLIBCXX_ABI_TAG_CXX11` constexpr `_Tp` `imag` () const
- constexpr void `imag` (`_Tp` \_\_val)
- constexpr `complex`< `_Tp` > & `operator*=` (const `_Tp` &)
- template<typename `_Up` >  
constexpr `complex`< `_Tp` > & `operator*=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator+=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
constexpr `complex`< `_Tp` > & `operator+=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator-=` (const `_Tp` &\_\_t)
- template<typename `_Up` >  
constexpr `complex`< `_Tp` > & `operator-=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator/=` (const `_Tp` &)
- template<typename `_Up` >  
constexpr `complex`< `_Tp` > & `operator/=` (const `complex`< `_Up` > &)
- constexpr `complex`< `_Tp` > & `operator=` (const `_Tp` &)
- constexpr `complex` & `operator=` (const `complex` &)=default
- template<typename `_Up` >  
constexpr `complex`< `_Tp` > & `operator=` (const `complex`< `_Up` > &)
- `_GLIBCXX_ABI_TAG_CXX11` constexpr `_Tp` `real` () const
- constexpr void `real` (`_Tp` \_\_val)

## 4.367.1 Detailed Description

```
template<typename _Tp>
struct std::complex< _Tp >
```

Template to represent complex numbers.

Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

## Parameters

|           |                                    |
|-----------|------------------------------------|
| <i>Tp</i> | Type of real and imaginary values. |
|-----------|------------------------------------|

Definition at line 67 of file complex.

## 4.367.2 Member Typedef Documentation

#### 4.367.2.1 value\_type

```
template<typename _Tp >
typedef _Tp std::complex< _Tp >::value_type
```

Value typedef.

Definition at line 130 of file complex.

### 4.367.3 Constructor & Destructor Documentation

#### 4.367.3.1 complex() [1/2]

```
template<typename _Tp >
constexpr std::complex< _Tp >::complex (
 const _Tp & __r = _Tp(),
 const _Tp & __i = _Tp()) [inline]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

Definition at line 134 of file complex.

#### 4.367.3.2 complex() [2/2]

```
template<typename _Tp >
template<typename _Up >
constexpr std::complex< _Tp >::complex (
 const complex< _Up > & __z) [inline]
```

Converting constructor.

Definition at line 144 of file complex.

### 4.367.4 Member Function Documentation

#### 4.367.4.1 operator+=()

```
template<typename _Tp >
constexpr complex<_Tp>& std::complex< _Tp >::operator+= (
 const _Tp & __t) [inline]
```

Add a scalar to this complex number.

Definition at line 189 of file complex.

#### 4.367.4.2 operator-=( )

```
template<typename _Tp >
constexpr complex<_Tp>& std::complex< _Tp >::operator-= (
 const _Tp & __t) [inline]
```

Subtract a scalar from this complex number.

Definition at line 198 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

### 4.368 [std::complex< double >](#) Struct Template Reference

#### Public Types

- typedef `__complex__ double` **\_ComplexT**
- typedef double **value\_type**

#### Public Member Functions

- constexpr **complex** (`_ComplexT __z`)
- constexpr **complex** (`double __r=0.0, double __i=0.0`)
- constexpr **complex** (`const complex< float > &__z`)
- constexpr **complex** (`const complex< long double > &`)
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr double `real()` const
- **\_\_attribute** (`((__abi_tag__("cxx11")))`) const expr double `imag()` const
- constexpr `_ComplexT __rep` () const
- constexpr void **imag** (`double __val`)
- constexpr [complex](#) & **operator\*=** (`double __d`)
- template<typename `_Tp` >  
constexpr [complex](#) & **operator\*=** (`const complex< _Tp > &__z`)
- constexpr [complex](#) & **operator+=** (`double __d`)
- template<typename `_Tp` >  
constexpr [complex](#) & **operator+=** (`const complex< _Tp > &__z`)
- constexpr [complex](#) & **operator-=** (`double __d`)
- template<typename `_Tp` >  
constexpr [complex](#) & **operator-=** (`const complex< _Tp > &__z`)
- constexpr [complex](#) & **operator/=** (`double __d`)
- template<typename `_Tp` >  
constexpr [complex](#) & **operator/=** (`const complex< _Tp > &__z`)
- constexpr [complex](#) & **operator=** (`double __d`)
- constexpr [complex](#) & **operator=** (`const complex &__z`)=default
- template<typename `_Tp` >  
constexpr [complex](#) & **operator=** (`const complex< _Tp > &__z`)
- constexpr void **real** (`double __val`)



## 4.368.1 Detailed Description

```
template<>
struct std::complex< double >
```

26.2.3 complex specializations complex<double> specialization

Definition at line 1227 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

## 4.369 std::complex&lt; float &gt; Struct Template Reference

## Public Types

- typedef \_\_complex\_\_ float **\_ComplexT**
- typedef float **value\_type**

## Public Member Functions

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (float \_\_r=0.0f, float \_\_i=0.0f)
- constexpr **complex** (const [complex](#)< double > &)
- constexpr **complex** (const [complex](#)< long double > &)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr float imag() const
- constexpr \_ComplexT **\_\_rep** () const
- constexpr void **imag** (float \_\_val)
- constexpr [complex](#) & **operator\*=** (float \_\_f)
- template<class \_Tp >  
constexpr [complex](#) & **operator\*=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator+=** (float \_\_f)
- template<typename \_Tp >  
constexpr [complex](#) & **operator+=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator-=** (float \_\_f)
- template<class \_Tp >  
constexpr [complex](#) & **operator-=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator/=** (float \_\_f)
- template<class \_Tp >  
constexpr [complex](#) & **operator/=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator=** (float \_\_f)
- constexpr [complex](#) & **operator=** (const [complex](#) &)=default
- template<typename \_Tp >  
constexpr [complex](#) & **operator=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr void **real** (float \_\_val)

#### 4.369.1 Detailed Description

```
template<>
struct std::complex< float >
```

#### 26.2.3 complex specializations complex<float> specialization

Definition at line 1082 of file complex.

The documentation for this struct was generated from the following file:

- [complex](#)

#### 4.370 std::complex< long double > Struct Template Reference

##### Public Types

- typedef \_\_complex\_\_ long double **\_ComplexT**
- typedef long double **value\_type**

##### Public Member Functions

- constexpr **complex** (\_ComplexT \_\_z)
- constexpr **complex** (long double \_\_r=0.0L, long double \_\_i=0.0L)
- constexpr **complex** (const [complex](#)< float > &\_\_z)
- constexpr **complex** (const [complex](#)< double > &\_\_z)
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double real() const
- **\_\_attribute** ((\_\_abi\_tag\_\_("cxx11"))) const expr long double imag() const
- constexpr \_ComplexT **\_\_rep** () const
- constexpr void **imag** (long double \_\_val)
- constexpr [complex](#) & **operator\*=** (long double \_\_r)
- template<typename \_Tp >  
constexpr [complex](#) & **operator\*=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator+=** (long double \_\_r)
- template<typename \_Tp >  
constexpr [complex](#) & **operator+=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator-=** (long double \_\_r)
- template<typename \_Tp >  
constexpr [complex](#) & **operator-=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator/=** (long double \_\_r)
- template<typename \_Tp >  
constexpr [complex](#) & **operator/=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr [complex](#) & **operator=** (long double \_\_r)
- constexpr [complex](#) & **operator=** (const [complex](#) &)=default
- template<typename \_Tp >  
constexpr [complex](#) & **operator=** (const [complex](#)< \_Tp > &\_\_z)
- constexpr void **real** (long double \_\_val)

## 4.370.1 Detailed Description

```
template<>
struct std::complex< long double >
```

26.2.3 complex specializations `complex<long double>` specialization

Definition at line 1372 of file `complex`.

The documentation for this struct was generated from the following file:

- [complex](#)

4.371 `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >` Class Template Reference

## Public Types

- typedef `HT_Map::entry` **entry**
- typedef `HT_Map::entry_allocator` **entry\_allocator**
- typedef `alloc_traits::allocator_type` **entry\_allocator**
- typedef `alloc_traits::pointer` **entry\_pointer**
- typedef `HT_Map::key_type` **key\_type**

## Public Member Functions

- **cond\_dealtor** (`entry_allocator *p_a, entry *p_e`)
- **cond\_dealtor** (`entry_pointer p_e`)
- void **set\_key\_destruct** ()
- void **set\_no\_action** ()
- void **set\_no\_action\_destructor** ()

## Protected Attributes

- bool **m\_key\_destruct**
- `entry_allocator *const` **m\_p\_a**
- `entry *const` **m\_p\_e**

## 4.371.1 Detailed Description

```
template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >
```

Conditional deallocate constructor argument.

Conditional dey destructor, `cc_hash`.

Definition at line 52 of file `cond_dealtor.hpp`.

The documentation for this class was generated from the following files:

- [cond\\_dealtor.hpp](#)
- [cond\\_key\\_dtor\\_entry\\_dealtor.hpp](#)

#### 4.372 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >` Class Template Reference

##### Public Member Functions

- **cond\_dtor** (value\_vector a\_vec, iterator &r\_last\_it, Size\_Type total\_size)
- void **set\_no\_action** ()

##### Protected Attributes

- value\_vector **m\_a\_vec**
- const Size\_Type **m\_max\_size**
- bool **m\_no\_action**
- iterator & **m\_r\_last\_it**

##### 4.372.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
```

Conditional destructor.

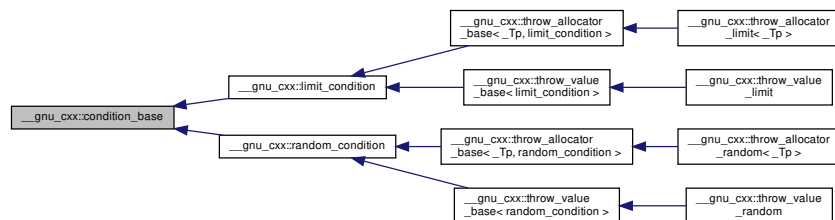
Definition at line 184 of file `ov_tree_map_.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_.hpp](#)

#### 4.373 `__gnu_cxx::condition_base` Struct Reference

Inheritance diagram for `__gnu_cxx::condition_base`:



## Public Member Functions

- **condition\_base** (const [condition\\_base](#) &)=default
- [condition\\_base](#) & **operator=** (const [condition\\_base](#) &)=default

## 4.373.1 Detailed Description

Base struct for condition policy.

Requires a public member function with the signature void throw\_conditionally()

Definition at line 414 of file throw\_allocator.h.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.374 std::condition\_variable Class Reference

## Public Types

- typedef \_\_native\_type \* **native\_handle\_type**

## Public Member Functions

- **condition\_variable** (const [condition\\_variable](#) &)=delete
- native\_handle\_type **native\_handle** ()
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable](#) & **operator=** (const [condition\\_variable](#) &)=delete
- void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock) noexcept
- template<typename \_Predicate >  
void **wait** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, \_Predicate \_\_p)
- template<typename \_Rep, typename \_Period >  
[cv\\_status](#) **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Rep, typename \_Period, typename \_Predicate >  
bool **wait\_for** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< system\_clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration >  
[cv\\_status](#) **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Clock, typename \_Duration, typename \_Predicate >  
bool **wait\_until** ([unique\\_lock](#)< [mutex](#) > &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

#### 4.374.1 Detailed Description

condition\_variable

Definition at line 71 of file condition\_variable.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

### 4.375 std::\_V2::condition\_variable\_any Class Reference

#### Public Member Functions

- **condition\_variable\_any** (const [condition\\_variable\\_any](#) &)=delete
- void **notify\_all** () noexcept
- void **notify\_one** () noexcept
- [condition\\_variable\\_any](#) & **operator=** (const [condition\\_variable\\_any](#) &)=delete
- template<typename \_Lock >  
void **wait** (\_Lock &\_\_lock)
- template<typename \_Lock , typename \_Predicate >  
void **wait** (\_Lock &\_\_lock, \_Predicate \_\_p)
- template<typename \_Lock , typename \_Rep , typename \_Period >  
[cv\\_status](#) **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Lock , typename \_Rep , typename \_Period , typename \_Predicate >  
bool **wait\_for** (\_Lock &\_\_lock, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime, \_Predicate \_\_p)
- template<typename \_Lock , typename \_Clock , typename \_Duration >  
[cv\\_status](#) **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Lock , typename \_Clock , typename \_Duration , typename \_Predicate >  
bool **wait\_until** (\_Lock &\_\_lock, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime, \_Predicate \_\_p)

#### 4.375.1 Detailed Description

condition\_variable\_any

Definition at line 253 of file condition\_variable.

The documentation for this class was generated from the following file:

- [condition\\_variable](#)

### 4.376 std::conditional< \_Cond, \_Iftrue, \_Iffalse > Struct Template Reference

#### Public Types

- typedef \_Iftrue **type**

## 4.377 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator` Struct Reference 2267

### 4.376.1 Detailed Description

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional< _Cond, _Iftrue, _Iffalse >
```

Define a member typedef `type` to one of two argument types.

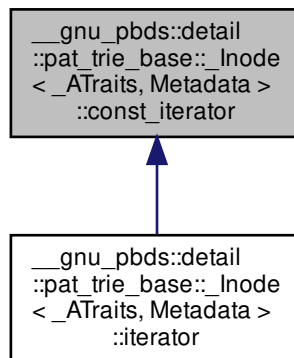
Definition at line 92 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.377 `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator` Struct Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_base::_Inode<_ATraits, Metadata>::const_iterator`:



### Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **const\_iterator** (node\_pointer\_pointer p\_p\_cur=0, node\_pointer\_pointer p\_p\_end=0)
- bool **operator!=** (const [const\\_iterator](#) &other) const
- node\_const\_pointer **operator\*** () const
- [const\\_iterator](#) & **operator++** ()
- [const\\_iterator](#) **operator++** (int)
- const node\_pointer\_pointer **operator->** () const
- bool **operator==** (const [const\\_iterator](#) &other) const

## Public Attributes

- node\_pointer\_pointer **m\_p\_p\_cur**
- node\_pointer\_pointer **m\_p\_p\_end**

## 4.377.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node< ATraits, Metadata >::const_iterator
```

Constant child iterator.

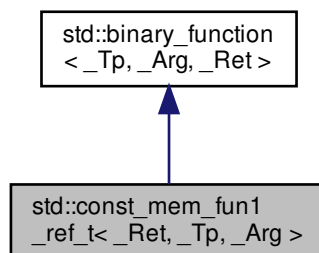
Definition at line 255 of file pat\_trie\_base.hpp.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.378 std::const\_mem\_fun1\_ref\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:





## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **const\_mem\_fun1\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)
- **\_Ret operator()** (const \_Tp &\_\_r, \_Arg \_\_x) const

### 4.378.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1329 of file stl\_function.h.

### 4.378.2 Member Typedef Documentation

#### 4.378.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]
```

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.378.2.2 result\_type

```
typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]
```

[result\\_type](#) is the return type

Definition at line 127 of file stl\_function.h.

#### 4.378.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

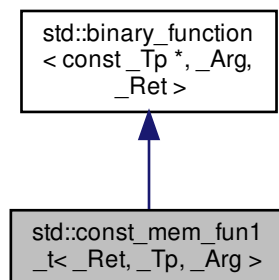
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.379 std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg > Class Template Reference

Inheritance diagram for std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



##### Public Types

- typedef const \_Tp \* [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

##### Public Member Functions

- **const\_mem\_fun1\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg) const)
- \_Ret **operator()** (const \_Tp \* \_\_p, \_Arg \_\_x) const

#### 4.379.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t<_Ret, _Tp, _Arg>
```

One of the [adaptors for member pointers](#).

Definition at line 1293 of file stl\_function.h.

#### 4.379.2 Member Typedef Documentation

##### 4.379.2.1 first\_argument\_type

```
typedef const _Tp * std::binary_function< const _Tp * , _Arg , _Ret >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 4.379.2.2 result\_type

```
typedef _Ret std::binary_function< const _Tp * , _Arg , _Ret >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

##### 4.379.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< const _Tp * , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

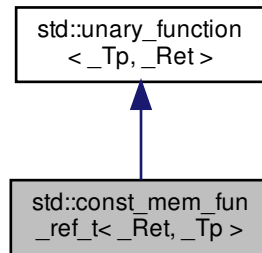
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 4.380 `std::const_mem_fun_ref_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::const_mem_fun_ref_t<_Ret, _Tp>`:



#### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

#### Public Member Functions

- **`const_mem_fun_ref_t`** (`_Ret(_Tp::*__pf)()` const)
- `_Ret` **`operator()`** (const `_Tp &__r`) const

#### 4.380.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1257 of file `stl_function.h`.

#### 4.380.2 Member Typedef Documentation

## 4.380.2.1 argument\_type

```
typedef _Tp std::unary_function< _Tp , _Ret >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 4.380.2.2 result\_type

```
typedef _Ret std::unary_function< _Tp , _Ret >::result_type [inherited]
```

result\_type is the return type

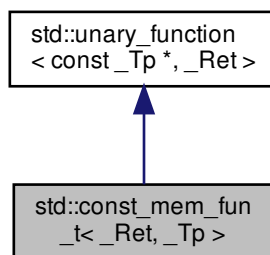
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.381 std::const\_mem\_fun\_t&lt;\_Ret, \_Tp&gt; Class Template Reference

Inheritance diagram for std::const\_mem\_fun\_t<\_Ret, \_Tp>:



## Public Types

- typedef const\_Tp \* [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

## Public Member Functions

- **const\_mem\_fun\_t** (\_Ret(\_Tp::\_\_pf)() const)
- **\_Ret operator()** (const \_Tp \*\_\_p) const

### 4.381.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::const_mem_fun_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1221 of file `stl_function.h`.

### 4.381.2 Member Typedef Documentation

#### 4.381.2.1 argument\_type

```
typedef const _Tp * std::unary_function< const _Tp * , _Ret >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.381.2.2 result\_type

```
typedef _Ret std::unary_function< const _Tp * , _Ret >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.382 `__gnu_cxx::constant_binary_fun<_Result, _Arg1, _Arg2>` Struct Template Reference

Inherits `__gnu_cxx::Constant_binary_fun<_Result, _Arg1, _Arg2>`.

## Public Types

- typedef `_Arg1` **first\_argument\_type**
- typedef `_Result` **result\_type**
- typedef `_Arg2` **second\_argument\_type**

## Public Member Functions

- **constant\_binary\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** (const `_Arg1` &, const `_Arg2` &) const

## Public Attributes

- `_Result` **M\_val**

## 4.382.1 Detailed Description

```
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
```

An [SGI extension](#) .

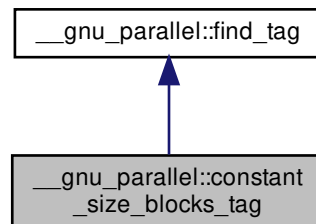
Definition at line 312 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

4.383 `__gnu_parallel::constant_size_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::constant_size_blocks_tag`:



#### 4.383.1 Detailed Description

Selects the constant block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`

Definition at line 178 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.384 `__gnu_cxx::constant_unary_fun<_Result, _Argument>` Struct Template Reference

Inherits `__gnu_cxx::_Constant_unary_fun<_Result, _Argument>`.

##### Public Types

- `typedef _Argument argument_type`
- `typedef _Result result_type`

##### Public Member Functions

- `constant_unary_fun (const _Result &__v)`
- `const result_type & operator() (const _Argument &) const`

##### Public Attributes

- `result_type _M_val`

#### 4.384.1 Detailed Description

```
template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun<_Result, _Argument>
```

An [SGL extension](#) .

Definition at line 304 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)



## 4.385 `__gnu_cxx::constant_void_fun<_Result>` Struct Template Reference

Inherits `__gnu_cxx::_Constant_void_fun<_Result>`.

### Public Types

- typedef `_Result` **result\_type**

### Public Member Functions

- **constant\_void\_fun** (const `_Result` &\_\_v)
- const `result_type` & **operator()** () const

### Public Attributes

- `result_type` **\_M\_val**

#### 4.385.1 Detailed Description

```
template<class _Result>
struct __gnu_cxx::constant_void_fun<_Result>
```

An [SGI extension](#) .

Definition at line 295 of file `ext/functional`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 4.386 `__gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_TI>` Struct Template Reference

#### 4.386.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_TI = null_type>
struct __gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, Tag, Policy_TI>
```

Dispatch mechanism, primary template for associative types.

Definition at line 449 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.387 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >` Struct Template Reference

##### Public Types

- typedef `binary_heap<_VTp, Cmp_Fn, _Alloc >` `type`

##### 4.387.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
```

Specialization for `binary_heap`.

Definition at line 95 of file `priority_queue_base_dispatch.hpp`.

##### 4.387.2 Member Typedef Documentation

###### 4.387.2.1 `type`

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp,
Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 99 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `priority_queue_base_dispatch.hpp`

#### 4.388 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >` Struct Template Reference

##### Public Types

- typedef `binomial_heap<_VTp, Cmp_Fn, _Alloc >` `type`

#### 4.388.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
```

Specialization for binomial\_heap.

Definition at line 77 of file priority\_queue\_base\_dispatch.hpp.

#### 4.388.2 Member Typedef Documentation

##### 4.388.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp,
Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 81 of file priority\_queue\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

## 4.389 \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >

### Struct Template Reference

#### Public Types

- typedef [pairing\\_heap](#)<\_VTp, Cmp\_Fn, \_Alloc > [type](#)

#### 4.389.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
```

Specialization for pairing\_heap.

Definition at line 68 of file priority\_queue\_base\_dispatch.hpp.

#### 4.389.2 Member Typedef Documentation

##### 4.389.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 72 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.390 [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch](#)< \_VTp, Cmp\_Fn, \_Alloc, [rc\\_binomial\\_heap\\_tag](#), [null\\_type](#) > Struct Template Reference

##### Public Types

- typedef [rc\\_binomial\\_heap](#)< \_VTp, Cmp\_Fn, \_Alloc > [type](#)

##### 4.390.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
```

Specialization for `rc_binary_heap`.

Definition at line 86 of file `priority_queue_base_dispatch.hpp`.

#### 4.390.2 Member Typedef Documentation

##### 4.390.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type
```

Dispatched type.

Definition at line 90 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [priority\\_queue\\_base\\_dispatch.hpp](#)

#### 4.391 `__gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>` Struct Template Reference

##### Public Types

- typedef `thin_heap<_VTp, Cmp_Fn, _Alloc>` `type`

##### 4.391.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>
```

Specialization for `thin_heap`.

Definition at line 104 of file `priority_queue_base_dispatch.hpp`.

##### 4.391.2 Member Typedef Documentation

###### 4.391.2.1 `type`

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch<_VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type>::type
```

Dispatched type.

Definition at line 108 of file `priority_queue_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `priority_queue_base_dispatch.hpp`

#### 4.392 `__gnu_pbds::detail::container_base_dispatch<Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl>` Struct Template Reference

##### Public Types

- typedef `cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t>` `type`

#### 4.392.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash map.

Definition at line 258 of file container\_base\_dispatch.hpp.

#### 4.392.2 Member Typedef Documentation

##### 4.392.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_d
Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 275 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.393 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_Tl > Struct Template Reference

##### Public Types

- typedef [gp\\_ht\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc, at3t::value, at4t, at5t, at2t > [type](#)

#### 4.393.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash map.

Definition at line 303 of file container\_base\_dispatch.hpp.

#### 4.393.2 Member Typedef Documentation

##### 4.393.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_b
Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 322 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.394 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >`

##### Struct Template Reference

##### Public Types

- typedef [lu\\_map](#)< Key, Mapped, at0t, \_Alloc, at1t > [type](#)

#### 4.394.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
```

Specialization for list-update map.

Definition at line 107 of file `container_base_dispatch.hpp`.

#### 4.394.2 Member Typedef Documentation

##### 4.394.2.1 type

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key,
Mapped, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 118 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.395 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef `ov_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

##### 4.395.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree map.

Definition at line 227 of file `container_base_dispatch.hpp`.

##### 4.395.2 Member Typedef Documentation

###### 4.395.2.1 `type`

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 237 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

#### 4.396 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef `pat_trie_map< Key, Mapped, at1t, _Alloc >` `type`



#### 4.396.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie map.

Definition at line 139 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

### 4.397 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >` Struct Template Reference

#### Public Types

- typedef [rb\\_tree\\_map](#)< Key, Mapped, at0t, at1t, \_Alloc > [type](#)

#### 4.397.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree map.

Definition at line 165 of file `container_base_dispatch.hpp`.

#### 4.397.2 Member Typedef Documentation

##### 4.397.2.1 `type`

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef rb_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 175 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.398 `__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef `splay_tree_map< Key, Mapped, at0t, at1t, _Alloc >` `type`

##### 4.398.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree map.

Definition at line 195 of file `container_base_dispatch.hpp`.

##### 4.398.2 Member Typedef Documentation

###### 4.398.2.1 `type`

```
template<typename Key , typename Mapped , typename _Alloc , typename Policy_Tl >
typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 206 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- `container_base_dispatch.hpp`

#### 4.399 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- typedef `cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t >` `type`

#### 4.399.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
```

Specialization collision-chaining hash set.

Definition at line 280 of file `container_base_dispatch.hpp`.

#### 4.399.2 Member Typedef Documentation

##### 4.399.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 298 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

## 4.400 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >` Struct Template Reference

### Public Types

- typedef `gp_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t >` type

#### 4.400.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
```

Specialization general-probe hash set.

Definition at line 327 of file `container_base_dispatch.hpp`.

#### 4.400.2 Member Typedef Documentation

##### 4.400.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 347 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.401 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI > Struct Template Reference

##### Public Types

- typedef lu\_set< Key, null\_type, at0t, \_Alloc, at1t > type

##### 4.401.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_TI>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_TI >
```

Specialization for list-update set.

Definition at line 123 of file container\_base\_dispatch.hpp.

#### 4.401.2 Member Typedef Documentation

##### 4.401.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 134 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.402 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- `typedef ov_tree_set< Key, null\_type, at0t, at1t, _Alloc > type`

##### 4.402.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
```

Specialization ordered-vector tree set.

Definition at line 242 of file `container_base_dispatch.hpp`.

##### 4.402.2 Member Typedef Documentation

###### 4.402.2.1 `type`

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 253 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.403 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- `typedef pat_trie_set< Key, null\_type, at1t, _Alloc > type`

#### 4.403.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
```

Specialization for PATRICIA trie set.

Definition at line 151 of file container\_base\_dispatch.hpp.

#### 4.403.2 Member Typedef Documentation

##### 4.403.2.1 type

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type
```

Dispatched type.

Definition at line 160 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.404 \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, rb\_tree\_tag, Policy\_Tl > Struct Template Reference

##### Public Types

- typedef rb\_tree\_set< Key, [null\\_type](#), at0t, at1t, \_Alloc > **type**

#### 4.404.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
```

Specialization for R-B tree set.

Definition at line 180 of file container\_base\_dispatch.hpp.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.405 `__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >` Struct Template Reference

##### Public Types

- `typedef splay_tree_set< Key, null\_type, at0t, at1t, _Alloc > type`

##### 4.405.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >
```

Specialization splay tree set.

Definition at line 211 of file `container_base_dispatch.hpp`.

##### 4.405.2 Member Typedef Documentation

###### 4.405.2.1 `type`

```
template<typename Key , typename _Alloc , typename Policy_Tl >
typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.

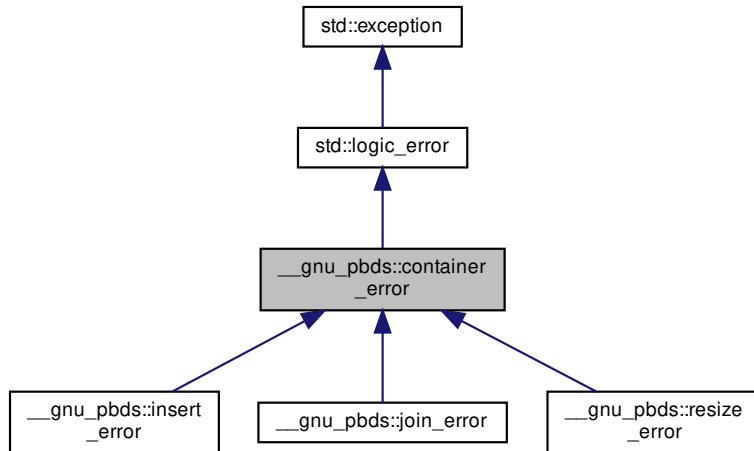
Definition at line 222 of file `container_base_dispatch.hpp`.

The documentation for this struct was generated from the following file:

- [container\\_base\\_dispatch.hpp](#)

#### 4.406 `__gnu_pbds::container_error` Struct Reference

Inheritance diagram for `__gnu_pbds::container_error`:



##### Public Member Functions

- virtual const char \* [what](#) () const noexcept

##### 4.406.1 Detailed Description

Base class for exceptions.

Definition at line 57 of file `exception.hpp`.

##### 4.406.2 Member Function Documentation

###### 4.406.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

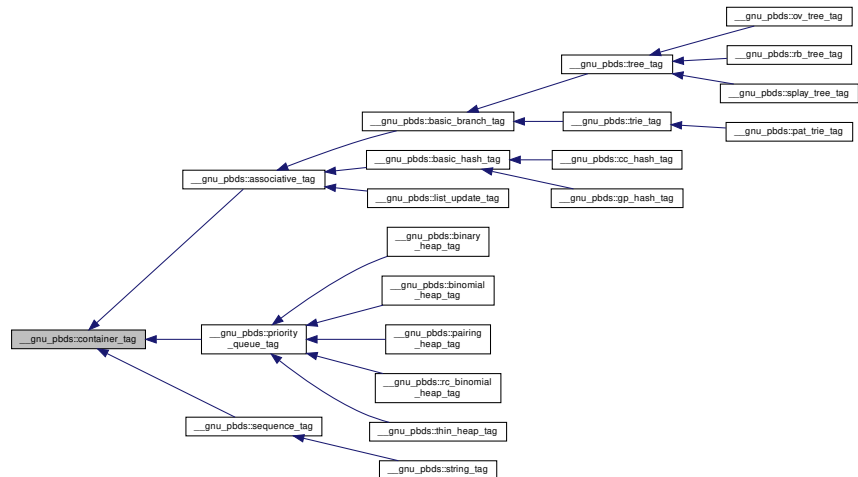
The documentation for this struct was generated from the following file:

- [exception.hpp](#)



## 4.407 \_\_gnu\_pbds::container\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::container\_tag:



## 4.407.1 Detailed Description

Base data structure tag.

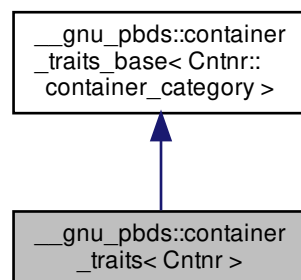
Definition at line 125 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.408 \_\_gnu\_pbds::container\_traits&lt; Cntnr &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::container\_traits< Cntnr >:



## Public Types

- enum { [order\\_preserving](#), [erase\\_can\\_throw](#), [split\\_join\\_can\\_throw](#), [reverse\\_iteration](#) }
- typedef [container\\_traits\\_base](#)< container\_category > **base\_type**
- typedef Cntnr::container\_category **container\_category**
- typedef Cntnr **container\_type**
- typedef base\_type::invalidation\_guarantee **invalidation\_guarantee**

### 4.408.1 Detailed Description

```
template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >
```

Container traits.

Definition at line 418 of file tag\_and\_trait.hpp.

### 4.408.2 Member Enumeration Documentation

#### 4.408.2.1 anonymous enum

```
template<typename Cntnr >
anonymous enum
```

#### Enumerator

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <a href="#">order_preserving</a>     | True only if Cntnr objects guarantee storing keys by order. |
| <a href="#">erase_can_throw</a>      | True only if erasing a key can throw.                       |
| <a href="#">split_join_can_throw</a> | True only if split or join operations can throw.            |
| <a href="#">reverse_iteration</a>    | True only reverse iterators are supported.                  |

Definition at line 426 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.409 \_\_gnu\_pbds::container\_traits\_base<\_Tag> Struct Template Reference

### 4.409.1 Detailed Description

```
template<typename _Tag>
struct __gnu_pbds::container_traits_base< _Tag >
```

Primary template, container traits base.

Definition at line 220 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.410 `__gnu_pbds::container_traits_base< binary_heap_tag >` Struct Template Reference

##### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `binary_heap_tag` `container_category`
- typedef `basic_invalidation_guarantee` `invalidation_guarantee`

##### 4.410.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binary_heap_tag >
```

Specialization, binary heap.

Definition at line 400 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.411 `__gnu_pbds::container_traits_base< binomial_heap_tag >` Struct Template Reference

##### Public Types

- enum { `order_preserving`, `erase_can_throw`, `split_join_can_throw`, `reverse_iteration` }
- typedef `binomial_heap_tag` `container_category`
- typedef `point_invalidation_guarantee` `invalidation_guarantee`

#### 4.411.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< binomial_heap_tag >
```

Specialization, binomial heap.

Definition at line 368 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.412 \_\_gnu\_pbds::container\_traits\_base< cc\_hash\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [cc\\_hash\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.412.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< cc_hash_tag >
```

Specialization, cc hash.

Definition at line 224 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.413 \_\_gnu\_pbds::container\_traits\_base< gp\_hash\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 4.413.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< gp_hash_tag >
```

Specialization, gp hash.

Definition at line 240 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.414 `__gnu_pbds::container_traits_base< list_update_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [list\\_update\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 4.414.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< list_update_tag >
```

Specialization, list update.

Definition at line 320 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.415 `__gnu_pbds::container_traits_base< ov_tree_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [ov\\_tree\\_tag](#) **container\_category**
- typedef [basic\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.415.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< ov_tree_tag >
```

Specialization, ov tree.

Definition at line 288 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.416 \_\_gnu\_pbds::container\_traits\_base< pairing\_heap\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [pairing\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.416.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pairing_heap_tag >
```

Specialization, pairing heap.

Definition at line 336 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.417 \_\_gnu\_pbds::container\_traits\_base< pat\_trie\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [pat\\_trie\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 4.417.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< pat_trie_tag >
```

Specialization, pat trie.

Definition at line 304 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.418 `__gnu_pbds::container_traits_base< rb_tree_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rb\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

## 4.418.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rb_tree_tag >
```

Specialization, rb tree.

Definition at line 256 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.419 `__gnu_pbds::container_traits_base< rc_binomial_heap_tag >` Struct Template Reference

## Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [rc\\_binomial\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.419.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
```

Specialization, rc binomial heap.

Definition at line 384 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.420 \_\_gnu\_pbds::container\_traits\_base< splay\_tree\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [splay\\_tree\\_tag](#) **container\_category**
- typedef [range\\_invalidation\\_guarantee](#) **invalidation\_guarantee**

#### 4.420.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< splay_tree_tag >
```

Specialization, splay tree.

Definition at line 272 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.421 \_\_gnu\_pbds::container\_traits\_base< thin\_heap\_tag > Struct Template Reference

##### Public Types

- enum { **order\_preserving**, **erase\_can\_throw**, **split\_join\_can\_throw**, **reverse\_iteration** }
- typedef [thin\\_heap\\_tag](#) **container\_category**
- typedef [point\\_invalidation\\_guarantee](#) **invalidation\_guarantee**



## 4.421.1 Detailed Description

```
template<>
struct __gnu_pbds::container_traits_base< thin_heap_tag >
```

Specialization, thin heap.

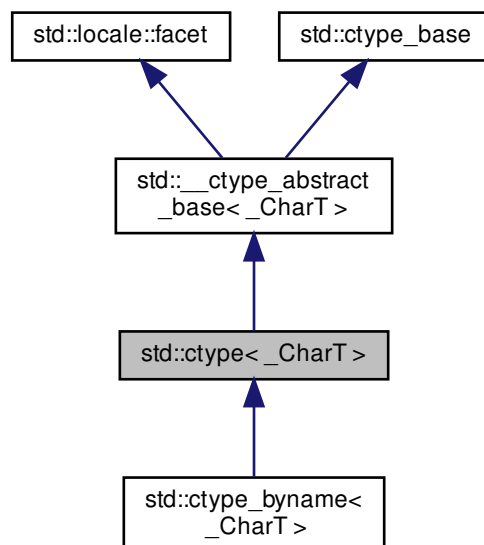
Definition at line 352 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.422 std::ctype&lt; \_CharT &gt; Class Template Reference

Inheritance diagram for std::ctype< \_CharT >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef [\\_\\_ctype\\_abstract\\_base< \\_CharT >::mask](#) **mask**

### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char\_type \_\_c) const
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const
- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \* \_\_vec) const
- virtual char **do\_narrow** (char\_type, char \_\_default) const
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_tolower** (char\_type \_\_c) const
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_toupper** (char\_type \_\_c) const
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_widen** (char \_\_c) const
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_dest) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.422.1 Detailed Description

```
template<typename _CharT>
class std::ctype<_CharT>
```

Primary class template ctype facet.

This template class defines classification and conversion functions for character sets. It wraps ctype functionality. Ctype gets used by streams for many I/O operations.

This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in `__ctype_abstract_base`, to allow for implementation flexibility. See `ctype<wchar_t>` for an example. The functions are documented in `__ctype_abstract_base`.

Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

Definition at line 612 of file `locale_facets.h`.

## 4.422.2 Member Function Documentation

## 4.422.2.1 do\_is() [1/2]

```
template<typename _CharT>
virtual bool std::ctype<_CharT>::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test `char_type` classification.

This function finds a mask `M` for `c` and compares it to mask `m`.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                                  |                                    |
|----------------------------------|------------------------------------|
| <a href="#"><code>__c</code></a> | The char_type to find the mask of. |
| <a href="#"><code>__m</code></a> | The mask to compare against.       |

**Returns**

(M & `__m`) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.422.2.2 do\_is()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype<_CharT>::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 4.422.2.3 do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype<_CharT>::do_narrow (
 char_type __c,
 char __dfault) const [protected], [virtual]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char_type to convert.           |
| <code>__dfault</code> | Char to return if conversion fails. |

## Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::narrow\(\)](#).

## 4.422.2.4 do\_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [protected], [virtual]
```

Narrow char\_type array to char.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__dfault` is used instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.422.2.5 do\_scan\_is()**

```
template<typename _CharT>
virtual const char_type* std::__ctype<_CharT>::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type matching mask.

This function searches for and returns the first char\_type c in [`__lo`,`__hi`) for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

**Parameters**

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 4.422.2.6 do\_scan\_not()

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find char\_type not matching mask.

This function searches for and returns a pointer to the first char\_type c of [lo,hi) for which is(m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a non-matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

## 4.422.2.7 do\_tolower() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**4.422.2.8 do\_tolower()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.422.2.9 do\_toupper()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_toupper (
 char_type __c) const [protected], [virtual]
```


Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.



## Parameters

|                                                                                                                      |                           |
|----------------------------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a><br><code>_c</code> | The char_type to convert. |
|----------------------------------------------------------------------------------------------------------------------|---------------------------|

## Returns

The uppercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

Referenced by `std::ctype< char >::toupper()`.

## 4.422.2.10 do\_toupper() [2/2]



```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

## Parameters

|                                                                                                                          |                            |
|--------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< \\_CharT >](#).

## 4.422.2.11 do\_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
 char __c) const [protected], [virtual]
```

Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                         |                      |
|-------------------------|----------------------|
| $\leftrightarrow$<br>_c | The char to convert. |
|-------------------------|----------------------|

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::widen()`.

#### 4.422.2.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::ctype<_CharT>::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [virtual]
```

Widen char array.

This function converts each char in the input to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                          |                                   |
|--------------------------|-----------------------------------|
| $\leftrightarrow$<br>_lo | Pointer to start range.           |
| $\leftrightarrow$<br>_hi | Pointer to end of range.          |
| $\leftrightarrow$<br>_to | Pointer to the destination array. |

**Returns**`__hi`.Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).**4.422.2.13 is()** [1/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↔type>::do_is()`.

**Parameters**

|                        |                                       |
|------------------------|---------------------------------------|
| <code>↔<br/>__c</code> | The char_type to compare the mask of. |
| <code>↔<br/>__m</code> | The mask to compare against.          |

**Returns**`(M & __m) != 0`.

Definition at line 169 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter>::get()`.**4.422.2.14 is()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Definition at line 186 of file `locale_facets.h`.

**4.422.2.15 narrow()** [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
 char_type __c,
 char __dfault) const [inline], [inherited]
```

Narrow `char_type` to `char`.

This function converts the `char_type` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. It does so by returning `ctype<char_type>::do_narrow(__c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                       |                                        |
|-----------------------|----------------------------------------|
| <code>__c</code>      | The <code>char_type</code> to convert. |
| <code>__dfault</code> | Char to return if conversion fails.    |

**Returns**

The converted `char`.

Definition at line 331 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _InIter >::get()`, and `std::time_put< _CharT, _OutIter >::put()`.

**4.422.2.16 narrow()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::narrow (
```

```

const char_type * __lo,
const char_type * __hi,
char __default,
char * __to) const [inline], [inherited]

```

Narrow array to char array.

This function converts each `char_type` in the input to `char` using the simplest reasonable transformation and writes the results to the destination array. For any `char_type` in the input that cannot be converted, *default* is used instead. It does so by returning `ctype<char_type>::do_narrow(__lo, __hi, __default, __to)`.

Note: this is not what you want for codepage conversions. See `codecv`t for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

#### Returns

`__hi`.

Definition at line 353 of file `locale_facets.h`.

#### 4.422.2.17 scan\_is()

```

template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]

```

Find `char_type` matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is true. It does so by returning `ctype<char_type>::do_scan_is()`.

#### Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 202 of file `locale_facets.h`.

**4.422.2.18 scan\_not()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

**Parameters**

|                                     |                                 |
|-------------------------------------|---------------------------------|
| <code>↔</code><br><code>__m</code>  | The mask to compare against.    |
| <code>↔</code><br><code>__lo</code> | Pointer to first char in range. |
| <code>↔</code><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

Definition at line 218 of file `locale_facets.h`.

**4.422.2.19 tolower()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

## Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else `__c`.

Definition at line 261 of file locale\_facets.h.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

## 4.422.2.20 tolower() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range `[__lo,__hi)` to lowercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_tolower(__lo,__hi)`.

## Parameters

|                   |                            |
|-------------------|----------------------------|
| <code>__lo</code> | Pointer to start of range. |
| <code>__hi</code> | Pointer to end of range.   |

## Returns

`__hi`.

Definition at line 276 of file locale\_facets.h.


## 4.422.2.21 toupper() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                                                                                                    |                           |
|----------------------------------------------------------------------------------------------------|---------------------------|
|  <code>__c</code> | The char_type to convert. |
|----------------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else `__c`.

Definition at line 232 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter >::get()`.



**4.422.2.22 toupper()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

**Parameters**

|                                                                                                       |                            |
|-------------------------------------------------------------------------------------------------------|----------------------------|
|  <code>__lo</code> | Pointer to start of range. |
|  <code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Definition at line 247 of file locale\_facets.h.

**4.422.2.23 widen()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT >::widen (
 char __c) const [inline], [inherited]
```



Widen `char` to `char_type`.

This function converts the `char` argument to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                 |                      |
|-----------------|----------------------|
| <code>_↵</code> | The char to convert. |
| <code>_c</code> |                      |

**Returns**

The converted `char_type`.

Definition at line 293 of file `locale_facets.h`.

Referenced by `std::time_get<_CharT, _InIter >::do_get()`, `std::money_get<_CharT, _InIter >::do_get()`, `std::time_↵put<_CharT, _OutIter >::do_put()`, and `std::money_put<_CharT, _OutIter >::do_put()`.

**4.422.2.24 widen() [2/2]**

```
template<typename _CharT>
const char* std::__ctype_abstract_base<_CharT >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to `char_type`.

This function converts each char in the input to `char_type` using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                     |                                   |
|-------------------------------------|-----------------------------------|
| <code>_↵</code><br><code>_lo</code> | Pointer to start of range.        |
| <code>_↵</code><br><code>_hi</code> | Pointer to end of range.          |
| <code>_↵</code><br><code>_to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 312 of file `locale_facets.h`.

**4.422.3 Member Data Documentation**

## 4.422.3.1 id

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id [static]
```

The facet id for ctype<char\_type>

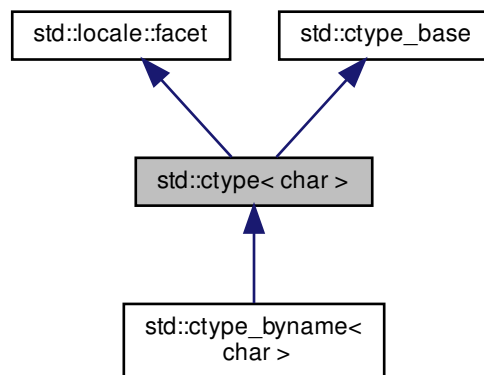
Definition at line 620 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

## 4.423 std::ctype&lt; char &gt; Class Template Reference

Inheritance diagram for std::ctype< char >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef char [char\\_type](#)
- typedef unsigned short **mask**

### Public Member Functions

- `ctype` (const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- `ctype` (\_\_c\_locale \_\_cloc, const mask \* \_\_table=0, bool \_\_del=false, size\_t \_\_refs=0)
- bool `is` (mask \_\_m, char \_\_c) const
- const char \* `is` (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char `narrow` (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* `narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- const char \* `scan_is` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* `scan_not` (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* `table` () const throw ()
- char\_type `tolower` (char\_type \_\_c) const
- const char\_type \* `tolower` (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type `toupper` (char\_type \_\_c) const
- const char\_type \* `toupper` (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type `widen` (char \_\_c) const
- const char \* `widen` (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

### Static Public Member Functions

- static const mask \* `classic_table` () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static `locale::id` id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t `table_size`
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual `~ctype` ()
- virtual char `do_narrow` (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* `do_narrow` (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \* \_\_to) const
- virtual char\_type `do_tolower` (char\_type \_\_c) const
- virtual const char\_type \* `do_tolower` (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type `do_toupper` (char\_type \_\_c) const
- virtual const char\_type \* `do_toupper` (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type `do_widen` (char \_\_c) const
- virtual const char \* `do_widen` (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_ctype**
- bool **\_M\_del**
- char **\_M\_narrow** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_narrow\_ok**
- const mask \* **\_M\_table**
- \_\_to\_type **\_M\_tolower**
- \_\_to\_type **\_M\_toupper**
- char **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- char **\_M\_widen\_ok**

## 4.423.1 Detailed Description

```
template<>
class std::ctype< char >
```

The ctype<char> specialization.

This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

Definition at line 681 of file locale\_facets.h.

## 4.423.2 Member Typedef Documentation

## 4.423.2.1 char\_type

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter char.

Definition at line 686 of file locale\_facets.h.

### 4.423.3 Constructor & Destructor Documentation

#### 4.423.3.1 ctype() [1/2]

```
std::ctype< char >::ctype (
 const mask * __table = 0,
 bool __del = false,
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                      |                                                                                             |
|----------------------|---------------------------------------------------------------------------------------------|
| <code>__table</code> | If non-zero, table is used as the per-char mask. Else <code>classic_table()</code> is used. |
| <code>__del</code>   | If true, passes ownership of table to this facet.                                           |
| <code>__refs</code>  | Passed to the base facet class.                                                             |

#### 4.423.3.2 ctype() [2/2]

```
std::ctype< char >::ctype (
 __c_locale __cloc,
 const mask * __table = 0,
 bool __del = false,
 size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

##### Parameters

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__cloc</code>  | Handle to C locale data.                          |
| <code>__table</code> | If non-zero, table is used as the per-char mask.  |
| <code>__del</code>   | If true, passes ownership of table to this facet. |
| <code>__refs</code>  | Passed to the base facet class.                   |

#### 4.423.3.3 ~ctype()

```
virtual std::ctype< char >::~ctype () [protected], [virtual]
```

Destructor.

This function deletes table() if *del* was true in the constructor.

#### 4.423.4 Member Function Documentation

##### 4.423.4.1 classic\_table()

```
static const mask* std::ctype< char >::classic_table () throw () [static]
```

Returns a pointer to the C locale mask table.

##### 4.423.4.2 do\_narrow() [1/2]

```
virtual char std::ctype< char >::do_narrow (
 char_type __c,
 char __dfault) const [inline], [protected], [virtual]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

##### Parameters

|                       |                                     |
|-----------------------|-------------------------------------|
| <code>__c</code>      | The char to convert.                |
| <code>__dfault</code> | Char to return if conversion fails. |

##### Returns

The converted char.

Definition at line 1134 of file locale\_facets.h.

#### 4.423.4.3 do\_narrow() [2/2]

```
virtual const char_type* std::ctype< char >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __dfault,
 char * __to) const [inline], [protected], [virtual]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

##### Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

##### Returns

`__hi`.

Definition at line 1160 of file locale\_facets.h.

#### 4.423.4.4 do\_tolower() [1/2]

```
virtual char_type std::ctype< char >::do_tolower (
 char_type __c) const [protected], [virtual]
```


Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.



## Parameters

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

## Returns

The lowercase char if convertible, else `__c`.

## 4.423.4.5 do\_tolower() [2/2]



```
virtual const char_type* std::ctype< char >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                                                                                     |                                 |
|-------------------------------------------------------------------------------------|---------------------------------|
|  | Pointer to first char in range. |
| <code>__lo</code>                                                                   |                                 |
|  | Pointer to end of range.        |
| <code>__hi</code>                                                                   |                                 |

## Returns

`__hi`.

## 4.423.4.6 do\_toupper() [1/2]


```
virtual char_type std::ctype< char >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

**Returns**

The uppercase char if convertible, else `__c`.

**4.423.4.7 do\_toupper()** [2/2]



```
virtual const char_type* std::ctype< char >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                          |                            |
|----------------------------------------------------------------------------------------------------------|----------------------------|
| <br><code>__lo</code> | Pointer to start of range. |
| <br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

**4.423.4.8 do\_widen()** [1/2]

```
virtual char_type std::ctype< char >::do_widen (
 char __c) const [inline], [protected], [virtual]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The converted character.

Definition at line 1084 of file locale\_facets.h.

## 4.423.4.9 do\_widen() [2/2]

```
virtual const char* std::ctype< char >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [protected], [virtual]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an undervied ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 1107 of file locale\_facets.h.

**4.423.4.10** `is()` [1/2]

```
bool std::ctype< char >::is (
 mask __m,
 char __c) const [inline]
```

Test char classification.

This function compares the mask `table[c]` to `__m`.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

**Returns**

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file `ctype_inline.h`.

**4.423.4.11** `is()` [2/2]

```
const char * std::ctype< char >::is (
 const char * __lo,
 const char * __hi,
 mask * __vec) const [inline]
```

Return a mask array.

This function finds the mask for each char in the range `[lo, hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Definition at line 48 of file `ctype_inline.h`.

## 4.423.4.12 narrow() [1/2]

```
char std::ctype< char >::narrow (
 char_type __c,
 char __default) const [inline]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, *default* is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(*c*). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char to convert.                |
| <code>__default</code> | Char to return if conversion fails. |

## Returns

The converted character.

Definition at line 931 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_narrow().

## 4.423.4.13 narrow() [2/2]

```
const char_type* std::ctype< char >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(*lo*, *hi*, *default*, *to*). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 964 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_narrow()`.

**4.423.4.14 `scan_is()`**

```
const char * std::ctype< char >::scan_is (
 mask __m,
 const char * __lo,
 const char * __hi) const [inline]
```

Find char matching a mask.

This function searches for and returns the first char in `[lo,hi)` for which `is(m,char)` is true.

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>↵<br/>__m</code>  | The mask to compare against. |
| <code>↵<br/>__lo</code> | Pointer to start of range.   |
| <code>↵<br/>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file `ctype_inline.h`.




## 4.423.4.15 scan\_not()

```
const char * std::ctype< char >::scan_not (
 mask __m,
 const char * __lo,
 const char * __hi) const [inline]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

## Parameters

|                                                                                           |                              |
|-------------------------------------------------------------------------------------------|------------------------------|
| <br>__m  | The mask to compare against. |
| <br>__lo | Pointer to start of range.   |
| <br>__hi | Pointer to end of range.     |

## Returns

Pointer to a non-matching char if found, else \_\_hi.

Definition at line 67 of file ctype\_inline.h.

## 4.423.4.16 table()

```
const mask* std::ctype< char >::table () const throw () [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

Definition at line 983 of file locale\_facets.h.

## 4.423.4.17 tolower() [1/2]


```
char_type std::ctype< char >::tolower (
 char_type __c) const [inline]
```

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_c). do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

**Returns**

The lowercase char if convertible, else `__c`.

Definition at line 835 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_tolower()`.

**4.423.4.18 tolower()** [2/2]



```
const char_type* std::ctype< char >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                                                                     |                                 |
|-------------------------------------------------------------------------------------|---------------------------------|
|  | Pointer to first char in range. |
| <code>__lo</code>                                                                   |                                 |
|  | Pointer to end of range.        |
| <code>__hi</code>                                                                   |                                 |

**Returns**

`__hi`.

Definition at line 852 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_tolower()`.



## 4.423.4.19 toupper() [1/2]

```
char_type std::ctype< char >::toupper (
 char_type __c) const [inline]
```

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do\_toupper(c). do\_toupper() must always return the same result for the same input.

**Parameters**

|                          |                      |
|--------------------------|----------------------|
| $\leftrightarrow$<br>__c | The char to convert. |
|--------------------------|----------------------|

**Returns**

The uppercase char if convertible, else \_\_c.

Definition at line 802 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_toupper().

## 4.423.4.20 toupper() [2/2]

```
const char_type* std::ctype< char >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline]
```

Convert array to uppercase.

This function converts each char in the range [\_\_lo,\_\_hi) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>:: do\_toupper(\_\_lo, \_\_hi). do\_toupper() must always return the same result for the same input.

**Parameters**

|                           |                                 |
|---------------------------|---------------------------------|
| $\leftrightarrow$<br>__lo | Pointer to first char in range. |
| $\leftrightarrow$<br>__hi | Pointer to end of range.        |

**Returns**

`__hi`.

Definition at line 819 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_toupper()`.

**4.423.4.21 widen()** [1/2]

```
char_type std::ctype< char >::widen (
 char __c) const [inline]
```

Widen char.

This function converts the `char` to `char_type` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted character.

Definition at line 872 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

**4.423.4.22 widen()** [2/2]

```
const char* std::ctype< char >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline]
```




Widen char array.

This function converts each `char` in the input to `char` using the simplest reasonable transformation. For an underived `ctype<char>` facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                                                                                                      |                                   |
|------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#">_lo</a> | Pointer to first char in range.   |
| <a href="#">_hi</a> | Pointer to end of range.          |
| <a href="#">_to</a> | Pointer to the destination array. |

## Returns

[!\[\]\(a03a7eb2f4046e1d3c76772003e549ea\_img.jpg\)\\_hi](#).

Definition at line 899 of file locale\_facets.h.

References `std::ctype<_CharT>::do_widen()`.

## 4.423.5 Member Data Documentation

## 4.423.5.1 id

```
locale::id std::ctype< char >::id [static]
```

The facet id for `ctype<char>`

Definition at line 703 of file locale\_facets.h.

## 4.423.5.2 table\_size

```
const size_t std::ctype< char >::table_size [static]
```

The size of the mask table. It is `SCHAR_MAX + 1`.

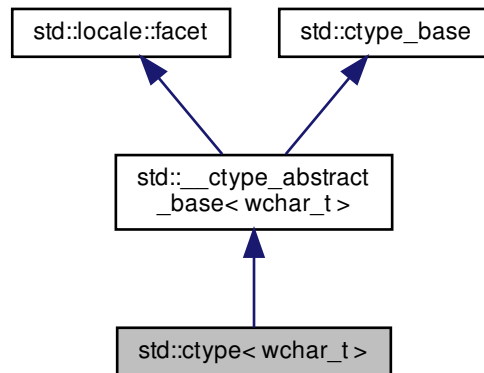
Definition at line 705 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [ctype\\_inline.h](#)

#### 4.424 std::ctype< wchar\_t > Class Template Reference

Inheritance diagram for std::ctype< wchar\_t >:



##### Public Types

- typedef const int \* **\_\_to\_type**
- typedef wctype\_t **\_\_wmask\_type**
- typedef wchar\_t **char\_type**
- typedef unsigned short **mask**

##### Public Member Functions

- **ctype** (size\_t \_\_refs=0)
- **ctype** (\_\_c\_locale \_\_cloc, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, **char\_type** \_\_c) const
- const **char\_type** \* **is** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (**char\_type** \_\_c, char \_\_default) const
- const **char\_type** \* **narrow** (const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi, char \_\_default, char \* \_\_to) const
- const **char\_type** \* **scan\_is** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- const **char\_type** \* **scan\_not** (mask \_\_m, const **char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **tolower** (**char\_type** \_\_c) const
- const **char\_type** \* **tolower** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **toupper** (**char\_type** \_\_c) const
- const **char\_type** \* **toupper** (**char\_type** \* \_\_lo, const **char\_type** \* \_\_hi) const
- **char\_type** **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, **char\_type** \* \_\_to) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual [~ctype](#) ()
- [\\_\\_wmask\\_type](#) **\_M\_convert\_to\_wmask** (const mask \_\_m) const throw ()
- void **\_M\_initialize\_ctype** () throw ()
- virtual bool [do\\_is](#) (mask \_\_m, [char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_is](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, mask \* \_\_vec) const
- virtual [char](#) [do\\_narrow](#) ([char\\_type](#) \_\_c, [char](#) \_\_dfault) const
- virtual const [char\\_type](#) \* [do\\_narrow](#) (const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi, [char](#) \_\_dfault, [char](#) \* \_\_to) const
- virtual const [char\\_type](#) \* [do\\_scan\\_is](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual const [char\\_type](#) \* [do\\_scan\\_not](#) (mask \_\_m, const [char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_tolower](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_tolower](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_toupper](#) ([char\\_type](#) \_\_c) const
- virtual const [char\\_type](#) \* [do\\_toupper](#) ([char\\_type](#) \* \_\_lo, const [char\\_type](#) \* \_\_hi) const
- virtual [char\\_type](#) [do\\_widen](#) ([char](#) \_\_c) const
- virtual const [char](#) \* [do\\_widen](#) (const [char](#) \* \_\_lo, const [char](#) \* \_\_hi, [char\\_type](#) \* \_\_to) const

## Static Protected Member Functions

- static [\\_\\_c\\_locale](#) **\_S\_clone\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc, const [char](#) \* \_\_s, [\\_\\_c\\_locale](#) \_\_old=0)
- static void **\_S\_destroy\_c\_locale** ([\\_\\_c\\_locale](#) & \_\_cloc)
- static [\\_\\_c\\_locale](#) **\_S\_get\_c\_locale** ()
- static const [char](#) \* **\_S\_get\_c\_name** () throw ()
- static [\\_\\_c\\_locale](#) **\_S\_lc\_ctype\_c\_locale** ([\\_\\_c\\_locale](#) \_\_cloc, const [char](#) \* \_\_s)

## Protected Attributes

- mask **\_M\_bit** [16]
- [\\_\\_c\\_locale](#) **\_M\_c\_locale\_ctype**
- [char](#) **\_M\_narrow** [128]
- bool **\_M\_narrow\_ok**
- [wint\\_t](#) **\_M\_widen** [1+static\_cast< unsigned char >(-1)]
- [\\_\\_wmask\\_type](#) **\_M\_wmask** [16]

#### 4.424.1 Detailed Description

```
template<>
class std::ctype< wchar_t >
```

The ctype<wchar\_t> specialization.

This class defines classification and conversion functions for the wchar\_t type. It gets used by wchar\_t streams for many I/O operations. The wchar\_t specialization provides a number of optimizations as well.

ctype<wchar\_t> inherits its public methods from \_\_ctype\_abstract\_base<wchar\_t>.

Definition at line 1186 of file locale\_facets.h.

#### 4.424.2 Member Typedef Documentation

##### 4.424.2.1 char\_type

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter wchar\_t.

Definition at line 1191 of file locale\_facets.h.

#### 4.424.3 Constructor & Destructor Documentation

##### 4.424.3.1 ctype() [1/2]

```
std::ctype< wchar_t >::ctype (
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

## 4.424.3.2 ctype() [2/2]

```
std::ctype< wchar_t >::ctype (
 __c_locale __cloc,
 size_t __refs = 0) [explicit]
```

Constructor performs static initialization.

This constructor is used to construct the initial C locale facet.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | Handle to C locale data.        |
| <code>__refs</code> | Passed to the base facet class. |

## 4.424.3.3 ~ctype()

```
virtual std::ctype< wchar_t >::~ctype () [protected], [virtual]
```

Destructor.

## 4.424.4 Member Function Documentation

## 4.424.4.1 do\_is() [1/2]

```
virtual bool std::ctype< wchar_t >::do_is (
 mask __m,
 char_type __c) const [protected], [virtual]
```

Test wchar\_t classification.

This function finds a mask *M* for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The wchar_t to find the mask of. |
| <code>__m</code> | The mask to compare against.     |

**Returns**

(M & \_\_m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.2 do\_is()** [2/2]

```
virtual const char_type* std::ctype< wchar_t >::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual]
```

Return a mask array.

This function finds the mask for each `wchar_t` in the range `[lo,hi)` and successively writes it to `vec`. `vec` must have as many elements as the input.

`do_is()` is a hook for a derived facet to change the behavior of classifying. `do_is()` must always return the same result for the same input.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.3 do\_narrow()** [1/2]

```
virtual char std::ctype< wchar_t >::do_narrow (
 char_type __c,
 char __dfault) const [protected], [virtual]
```

Narrow `wchar_t` to `char`.

This virtual function converts the argument to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<wchar_t>` facet, `c` will be cast to `char` and returned.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.



## Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <code>__c</code>       | The <code>wchar_t</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.  |

## Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

## 4.424.4.4 do\_narrow() [2/2]

```
virtual const char_type* std::ctype< wchar_t >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual]
```

Narrow `wchar_t` array to char array.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to char using the simplest reasonable transformation and writes the results to the destination array. For any `wchar_t` in the input that cannot be converted, `default` is used instead. For an underived `ctype<wchar_t>` facet, the argument will be copied, casting each element to char.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

## 4.424.4.5 do\_scan\_is()

```
virtual const char_type* std::ctype< wchar_t >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t matching mask.

This function searches for and returns the first wchar\_t c in [\_\_lo,\_\_hi) for which is(\_\_m,c) is true.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching wchar\_t if found, else \_\_hi.

Implements `std::__ctype_abstract_base< wchar_t >`.

## 4.424.4.6 do\_scan\_not()

```
virtual const char_type* std::ctype< wchar_t >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Find wchar\_t not matching mask.

This function searches for and returns a pointer to the first wchar\_t c of [\_\_lo,\_\_hi) for which is(\_\_m,c) is false.

do\_scan\_is() is a hook for a derived facet to change the behavior of match searching. do\_is() must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching wchar\_t if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.7 do\_tolower()** [1/2]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
 char_type __c) const [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__c</code> | The wchar_t to convert. |
| <code>__C</code> |                         |

**Returns**

The lowercase wchar\_t if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.8 do\_tolower()** [2/2]



```
virtual const char_type* std::ctype< wchar_t >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each wchar\_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.

`do_tolower()` is a hook for a derived facet to change the behavior of lowercasing. `do_tolower()` must always return the same result for the same input.

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.9 do\_toupper()** [1/2]


```
virtual char_type std::ctype< wchar_t >::do_toupper (
 char_type __c) const [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the `wchar_t` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                         |                                      |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| <a href="#"></a><br><code>__c</code> | The <code>wchar_t</code> to convert. |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------|

**Returns**

The uppercase `wchar_t` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.10 do\_toupper()** [2/2]



```
virtual const char_type* std::ctype< wchar_t >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual]
```

Convert array to uppercase.

This virtual function converts each `wchar_t` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                        |                            |
|------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.11 do\_widen()** [1/2]

```
virtual char_type std::ctype< wchar_t >::do_widen (
 char __c) const [protected], [virtual]
```


Widen char to `wchar_t`.

This virtual function converts the char to `wchar_t` using the simplest reasonable transformation. For an underived `ctype<wchar_t>` facet, the argument will be cast to `wchar_t`.

`do_widen()` is a hook for a derived facet to change the behavior of widening. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                                                                                                                         |                      |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>__c</code> | The char to convert. |
|-------------------------------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The converted `wchar_t`.

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

**4.424.4.12 do\_widen()** [2/2]

```
virtual const char* std::ctype< wchar_t >::do_widen (
 const char * __lo,
```

```
const char * __hi,
char_type * __to) const [protected], [virtual]
```

Widen char array to wchar\_t array.

This function converts each char in the input to wchar\_t using the simplest reasonable transformation. For an underived ctype<wchar\_t> facet, the argument will be copied, casting each element to wchar\_t.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                   |                                   |
|-----------------------------------|-----------------------------------|
| <a href="#"><code>__lo</code></a> | Pointer to start range.           |
| <a href="#"><code>__hi</code></a> | Pointer to end of range.          |
| <a href="#"><code>__to</code></a> | Pointer to the destination array. |

#### Returns

[`\_\_hi`](#).

Implements [std::\\_\\_ctype\\_abstract\\_base< wchar\\_t >](#).

#### 4.424.4.13 is() [1/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test char\_type classification.

This function finds a mask M for `__c` and compares it to mask `__m`. It does so by returning the value of `ctype<char_↵ type>::do_is()`.

#### Parameters

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <a href="#"><code>__c</code></a> | The char_type to compare the mask of. |
| <a href="#"><code>__m</code></a> | The mask to compare against.          |

**Returns**

(M & \_\_m) != 0.

Definition at line 169 of file locale\_facets.h.

**4.424.4.14 is()** [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char\_type>::do\_is().

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

**Returns**

`__hi`.

Definition at line 186 of file locale\_facets.h.

**4.424.4.15 narrow()** [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |



**Returns**

The converted char.

Definition at line 331 of file locale\_facets.h.

**4.424.4.16 narrow()** [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, *default* is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 353 of file locale\_facets.h.




**4.424.4.17 scan\_is()**

```
const char_type* std::__ctype_abstract_base< wchar_t >::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                                                                                                                        |                              |
|------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against. |
| <a href="#"></a><br><code>__lo</code> | Pointer to start of range.   |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to matching `char_type` if found, else `__hi`.

Definition at line 202 of file `locale_facets.h`.




**4.424.4.18 scan\_not()**

```
const char_type* std::__ctype_abstract_base< wchar_t >::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find `char_type` not matching a mask.

This function searches for and returns the first `char_type` `c` in `[lo,hi)` for which `is(m,c)` is false. It does so by returning `ctype<char_type>::do_scan_not()`.

**Parameters**

|                                                                                                                          |                                 |
|--------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| <a href="#"></a><br><code>__m</code>  | The mask to compare against.    |
| <a href="#"></a><br><code>__lo</code> | Pointer to first char in range. |
| <a href="#"></a><br><code>__hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

Definition at line 218 of file `locale_facets.h`.

## 4.424.4.19 tolower() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_tolower(c).

## Parameters

|                     |                           |
|---------------------|---------------------------|
| $\leftarrow$<br>__c | The char_type to convert. |
|---------------------|---------------------------|

## Returns

The lowercase char\_type if convertible, else \_\_c.

Definition at line 261 of file locale\_facets.h.

## 4.424.4.20 tolower() [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo,\_\_hi).

## Parameters

|                      |                            |
|----------------------|----------------------------|
| $\leftarrow$<br>__lo | Pointer to start of range. |
| $\leftarrow$<br>__hi | Pointer to end of range.   |

## Returns

\_\_hi.

Definition at line 276 of file locale\_facets.h.


**4.424.4.21 toupper()** [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char\_type>::do\_toupper().

**Parameters**

|                                                                                          |                           |
|------------------------------------------------------------------------------------------|---------------------------|
| <br>__c | The char_type to convert. |
|------------------------------------------------------------------------------------------|---------------------------|

**Returns**

The uppercase char\_type if convertible, else \_\_c.

Definition at line 232 of file locale\_facets.h.



**4.424.4.22 toupper()** [2/2]

```
const char_type* std::__ctype_abstract_base< wchar_t >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char\_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_toupper(lo, hi).

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

Definition at line 247 of file locale\_facets.h.

## 4.424.4.23 widen() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (
 char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

## Returns

The converted char\_type.

Definition at line 293 of file locale\_facets.h.

## 4.424.4.24 widen() [2/2]

```
const char* std::__ctype_abstract_base< wchar_t >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                   |                                   |
|-------------------|-----------------------------------|
| <code>__lo</code> | Pointer to start of range.        |
| <code>__hi</code> | Pointer to end of range.          |
| <code>__to</code> | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 312 of file `locale_facets.h`.

## 4.424.5 Member Data Documentation

4.424.5.1 `id`

```
locale::id std::ctype< wchar_t >::id [static]
```

The facet id for `ctype<wchar_t>`

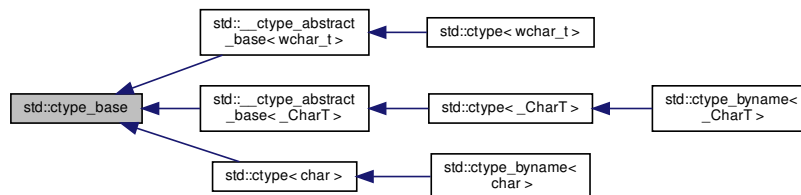
Definition at line 1209 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

4.425 `std::ctype_base` Struct Reference

Inheritance diagram for `std::ctype_base`:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef unsigned short **mask**

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## 4.425.1 Detailed Description

Base class for ctype.

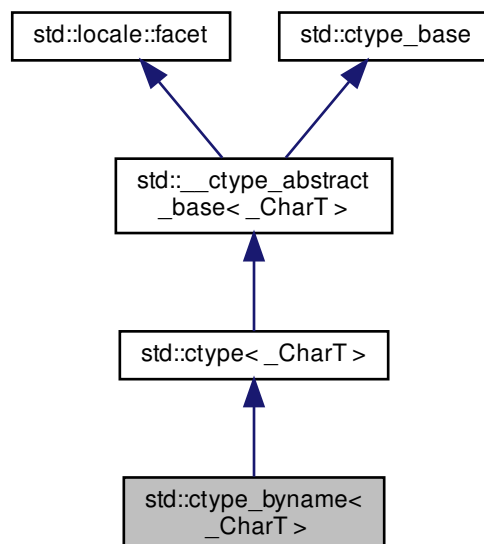
Definition at line 41 of file ctype\_base.h.

The documentation for this struct was generated from the following file:

- [ctype\\_base.h](#)

## 4.426 std::ctype\_byname&lt;\_CharT&gt; Class Template Reference

Inheritance diagram for std::ctype\_byname<\_CharT>:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef \_CharT **char\_type**
- typedef ctype< \_CharT >::mask **mask**

## Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **ctype\_byname** (const string & \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char\_type \_\_c) const
- const char\_type \* **is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \*\_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_dfault) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \*\_\_to) const
- const char\_type \* **scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- const char\_type \* **scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \*\_\_to) const

## Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id **id**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

## Protected Member Functions

- virtual bool **do\_is** (mask \_\_m, char\_type \_\_c) const
- virtual const char\_type \* **do\_is** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, mask \*\_\_vec) const
- virtual char **do\_narrow** (char\_type, char \_\_dfault) const
- virtual const char\_type \* **do\_narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual const char\_type \* **do\_scan\_is** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual const char\_type \* **do\_scan\_not** (mask \_\_m, const char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_tolower** (char\_type \_\_c) const
- virtual const char\_type \* **do\_tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_toupper** (char\_type \_\_c) const
- virtual const char\_type \* **do\_toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- virtual char\_type **do\_widen** (char \_\_c) const
- virtual const char \* **do\_widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \*\_\_dest) const



## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.426.1 Detailed Description

```
template<typename _CharT>
class std::ctype_byname<_CharT>
```

class ctype\_byname [22.2.1.2].

Definition at line 1478 of file locale\_facets.h.

## 4.426.2 Member Function Documentation

## 4.426.2.1 do\_is() [1/2]

```
template<typename _CharT>
virtual bool std::ctype<_CharT>::do_is (
 mask __m,
 char_type __c) const [protected], [virtual], [inherited]
```

Test char\_type classification.

This function finds a mask M for *c* and compares it to mask *m*.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

## Parameters

|                  |                                    |
|------------------|------------------------------------|
| <code>__c</code> | The char_type to find the mask of. |
| <code>__m</code> | The mask to compare against.       |

## Returns

(M & \_\_m) != 0.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 4.426.2.2 do\_is() [2/2]

```
template<typename _CharT>
virtual const char_type* std::__ctype<_CharT>::do_is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do\_is() is a hook for a derived facet to change the behavior of classifying. do\_is() must always return the same result for the same input.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

##### Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 4.426.2.3 do\_narrow() [1/2]

```
template<typename _CharT>
virtual char std::__ctype<_CharT>::do_narrow (
 char_type __c,
 char __dfault) const [protected], [virtual], [inherited]
```

Narrow char\_type to char.

This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do\_narrow() is a hook for a derived facet to change the behavior of narrowing. do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                        |                                        |
|------------------------|----------------------------------------|
| <code>__c</code>       | The <code>char_type</code> to convert. |
| <code>__default</code> | Char to return if conversion fails.    |

## Returns

The converted char.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::narrow()`.

## 4.426.2.4 do\_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [protected], [virtual], [inherited]
```

Narrow `char_type` array to `char`.

This virtual function converts each `char_type` in the range `[__lo,__hi)` to `char` using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, `__default` is used instead.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

## Returns

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

4.426.2.5 `do_scan_is()`

```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find `char_type` matching mask.

This function searches for and returns the first `char_type` `c` in `[__lo,__hi)` for which `is(__m,c)` is true.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching `char_type` if found, else `__hi`.

Implements `std::__ctype_abstract_base< _CharT >`.

4.426.2.6 `do_scan_not()`

```
template<typename _CharT>
virtual const char_type* std::ctype< _CharT >::do_scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Find `char_type` not matching mask.

This function searches for and returns a pointer to the first `char_type` `c` of `[lo,hi)` for which `is(m,c)` is false.

`do_scan_is()` is a hook for a derived facet to change the behavior of match searching. `do_is()` must always return the same result for the same input.

## Parameters

|                   |                              |
|-------------------|------------------------------|
| <code>__m</code>  | The mask to compare against. |
| <code>__lo</code> | Pointer to start of range.   |
| <code>__hi</code> | Pointer to end of range.     |

**Returns**

Pointer to a non-matching char\_type if found, else `__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.426.2.7 do\_tolower()** [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_tolower (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                  |                           |
|------------------|---------------------------|
| <code>__c</code> | The char_type to convert. |
|------------------|---------------------------|

**Returns**

The lowercase char\_type if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::tolower()`.

**4.426.2.8 do\_tolower()** [2/2]



```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char\_type in the range [`__lo`,`__hi`) to lowercase if possible. Other elements remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

**Parameters**

|                                                                                                                       |                            |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>_lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>_hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

**4.426.2.9 do\_toupper()** [1/2]


```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_toupper (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the `char_type` argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

`do_toupper()` is a hook for a derived facet to change the behavior of uppercasing. `do_toupper()` must always return the same result for the same input.

**Parameters**

|                                                                                                                        |                                        |
|------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <a href="#"></a><br><code>_c</code> | The <code>char_type</code> to convert. |
|------------------------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by `std::ctype<char>::toupper()`.

**4.426.2.10 do\_toupper()** [2/2]

```
template<typename _CharT>
virtual const char_type* std::ctype<_CharT>::do_toupper (
```



```
char_type * __lo,
const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char\_type in the range [\_\_lo,\_\_hi) to uppercase if possible. Other elements remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

#### Parameters

|                                                                                                        |                            |
|--------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"> __lo</a> | Pointer to start of range. |
| <a href="#"> __hi</a> | Pointer to end of range.   |

#### Returns

[!\[\]\(3cb60d42b10e53f9522bb0b392c1c4cd\_img.jpg\) \\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 4.426.2.11 do\_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype<_CharT>::do_widen (
 char __c) const [protected], [virtual], [inherited]
```


Widen char.

This virtual function converts the char to char\_type using the simplest reasonable transformation.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                         |                      |
|---------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"> __c</a> | The char to convert. |
|---------------------------------------------------------------------------------------------------------|----------------------|

#### Returns

The converted char\_type

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

Referenced by [std::ctype<char>::widen\(\)](#).

#### 4.426.2.12 do\_widen() [2/2]

```
template<typename _CharT>
virtual const char* std::ctype<_CharT>::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the input to [char\\_type](#) using the simplest reasonable transformation.

[do\\_widen\(\)](#) is a hook for a derived facet to change the behavior of widening. [do\\_widen\(\)](#) must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See [codecvt](#) for that.

##### Parameters

|                            |                                   |
|----------------------------|-----------------------------------|
| <a href="#">_↵<br/>_lo</a> | Pointer to start range.           |
| <a href="#">_↵<br/>_hi</a> | Pointer to end of range.          |
| <a href="#">_↵<br/>_to</a> | Pointer to the destination array. |

##### Returns

[\\_\\_hi](#).

Implements [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#).

#### 4.426.2.13 is() [1/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base<_CharT>::is (
 mask __m,
 char_type __c) const [inline], [inherited]
```

Test [char\\_type](#) classification.

This function finds a mask M for [\\_\\_c](#) and compares it to mask [\\_\\_m](#). It does so by returning the value of [ctype<char\\_↵  
type>::do\\_is\(\)](#).



## Parameters

|                  |                                       |
|------------------|---------------------------------------|
| <code>__c</code> | The char_type to compare the mask of. |
| <code>__m</code> | The mask to compare against.          |

## Returns

`(M & __m) != 0.`

Definition at line 169 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter>::get()`.

## 4.426.2.14 is() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::is (
 const char_type * __lo,
 const char_type * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char\_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of `ctype<char_type>::do_is()`.

## Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |

## Returns

`__hi.`

Definition at line 186 of file locale\_facets.h.

## 4.426.2.15 narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base<_CharT>::narrow (
```

```
char_type __c,
char __default) const [inline], [inherited]
```

Narrow char\_type to char.

This function converts the char\_type to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char_type to convert.           |
| <code>__default</code> | Char to return if conversion fails. |

#### Returns

The converted char.

Definition at line 331 of file locale\_facets.h.

Referenced by std::time\_get<\_CharT, \_InIter >::get(), and std::time\_put<\_CharT, \_OutIter >::put().

#### 4.426.2.16 narrow() [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow array to char array.

This function converts each char\_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char\_type in the input that cannot be converted, default is used instead. It does so by returning ctype<char\_type>::do\_narrow(\_\_lo, \_\_hi, \_\_default, \_\_to).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**`__hi`.

Definition at line 353 of file locale\_facets.h.

**4.426.2.17 scan\_is()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_is (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char\_type>::do\_scan\_is().

**Parameters**

|                         |                              |
|-------------------------|------------------------------|
| <code>↔<br/>__m</code>  | The mask to compare against. |
| <code>↔<br/>__lo</code> | Pointer to start of range.   |
| <code>↔<br/>__hi</code> | Pointer to end of range.     |

**Returns**Pointer to matching char\_type if found, else `__hi`.

Definition at line 202 of file locale\_facets.h.




**4.426.2.18 scan\_not()**

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base<_CharT>::scan_not (
 mask __m,
 const char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Find char\_type not matching a mask.

This function searches for and returns the first char\_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char\_type>::do\_scan\_not().

**Parameters**

|                                                                                                       |                                 |
|-------------------------------------------------------------------------------------------------------|---------------------------------|
| <br><code>_m</code>  | The mask to compare against.    |
| <br><code>_lo</code> | Pointer to first char in range. |
| <br><code>_hi</code> | Pointer to end of range.        |

**Returns**

Pointer to non-matching char if found, else `__hi`.

Definition at line 218 of file `locale_facets.h`.


**4.426.2.19 tolower()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_tolower(c)`.

**Parameters**

|                                                                                                        |                                        |
|--------------------------------------------------------------------------------------------------------|----------------------------------------|
| <br><code>_c</code> | The <code>char_type</code> to convert. |
|--------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The lowercase `char_type` if convertible, else `__c`.

Definition at line 261 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _InIter >::get()`.

**4.426.2.20 tolower()** [2/2]



```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::tolower (
```

```
char_type * __lo,
const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char\_type in the range [\_\_lo,\_\_hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char\_type>::do\_tolower(\_\_lo, \_\_hi).

**Parameters**

|                                                                                                        |                            |
|--------------------------------------------------------------------------------------------------------|----------------------------|
| <br><code>__lo</code> | Pointer to start of range. |
| <br><code>__hi</code> | Pointer to end of range.   |

**Returns**

`__hi`.

Definition at line 276 of file `locale_facets.h`.


**4.426.2.21 toupper()** [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning `ctype<char_type>::do_toupper()`.

**Parameters**

|                                                                                                         |                                        |
|---------------------------------------------------------------------------------------------------------|----------------------------------------|
| <br><code>__c</code> | The <code>char_type</code> to convert. |
|---------------------------------------------------------------------------------------------------------|----------------------------------------|

**Returns**

The uppercase `char_type` if convertible, else `__c`.

Definition at line 232 of file `locale_facets.h`.

Referenced by `std::time_get< _CharT, _Inlter >::get()`.



**4.426.2.22 toupper()** [2/2]

```
template<typename _CharT>
const char_type* std::__ctype_abstract_base< _CharT >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each `char_type` in the range `[lo,hi)` to uppercase if possible. Other elements remain untouched. It does so by returning `ctype<char_type>::do_toupper(lo, hi)`.

## Parameters

|                                                                                                                       |                            |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------|
| <a href="#"></a><br><code>_lo</code> | Pointer to start of range. |
| <a href="#"></a><br><code>_hi</code> | Pointer to end of range.   |

## Returns

`__hi.`

Definition at line 247 of file locale\_facets.h.

## 4.426.2.23 widen() [1/2]


```
template<typename _CharT>
char_type std::__ctype_abstract_base<_CharT>::widen (
 char __c) const [inline], [inherited]
```

Widen char to char\_type.

This function converts the char argument to char\_type using the simplest reasonable transformation. It does so by returning `ctype<char_type>::do_widen(c)`.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

## Parameters

|                                                                                                                        |                      |
|------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#"></a><br><code>_c</code> | The char to convert. |
|------------------------------------------------------------------------------------------------------------------------|----------------------|

## Returns

The converted char\_type.

Definition at line 293 of file locale\_facets.h.

Referenced by `std::time_get<_CharT, _InIter>::do_get()`, `std::money_get<_CharT, _InIter>::do_get()`, `std::time_put<_CharT, _OutIter>::do_put()`, and `std::money_put<_CharT, _OutIter>::do_put()`.

## 4.426.2.24 widen() [2/2]

```
template<typename _CharT>
const char* std::__ctype_abstract_base<_CharT>::widen (
```




```
const char * __lo,
const char * __hi,
char_type * __to) const [inline], [inherited]
```

Widen array to char\_type.

This function converts each char in the input to char\_type using the simplest reasonable transformation. It does so by returning ctype<char\_type>::do\_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

#### Parameters

|                                                                                                      |                                   |
|------------------------------------------------------------------------------------------------------|-----------------------------------|
| <a href="#">_lo</a> | Pointer to start of range.        |
| <a href="#">_hi</a> | Pointer to end of range.          |
| <a href="#">_to</a> | Pointer to the destination array. |

#### Returns

[!\[\]\(e8fb589d58dad1692debababa5e928b6\_img.jpg\)\\_hi](#).

Definition at line 312 of file locale\_facets.h.

### 4.426.3 Member Data Documentation

#### 4.426.3.1 id

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id [static], [inherited]
```

The facet id for ctype<char\_type>

Definition at line 620 of file locale\_facets.h.

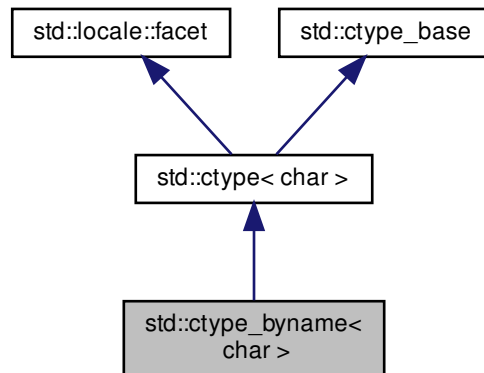
The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)



## 4.427 std::ctype\_byname&lt; char &gt; Class Template Reference

Inheritance diagram for std::ctype\_byname< char >:



## Public Types

- typedef const int \* **\_\_to\_type**
- typedef char **char\_type**
- typedef unsigned short **mask**

## Public Member Functions

- **ctype\_byname** (const char \* \_\_s, size\_t \_\_refs=0)
- **ctype\_byname** (const string & \_\_s, size\_t \_\_refs=0)
- bool **is** (mask \_\_m, char \_\_c) const
- const char \* **is** (const char \* \_\_lo, const char \* \_\_hi, mask \* \_\_vec) const
- char **narrow** (char\_type \_\_c, char \_\_default) const
- const char\_type \* **narrow** (const char\_type \* \_\_lo, const char\_type \* \_\_hi, char \_\_default, char \* \_\_to) const
- const char \* **scan\_is** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const char \* **scan\_not** (mask \_\_m, const char \* \_\_lo, const char \* \_\_hi) const
- const mask \* **table** () const throw ()
- char\_type **tolower** (char\_type \_\_c) const
- const char\_type \* **tolower** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **toupper** (char\_type \_\_c) const
- const char\_type \* **toupper** (char\_type \* \_\_lo, const char\_type \* \_\_hi) const
- char\_type **widen** (char \_\_c) const
- const char \* **widen** (const char \* \_\_lo, const char \* \_\_hi, char\_type \* \_\_to) const

### Static Public Member Functions

- static const mask \* [classic\\_table](#) () throw ()

### Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static [locale::id](#) id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size\_t [table\\_size](#)
- static const mask **upper**
- static const mask **xdigit**

### Protected Member Functions

- virtual char [do\\_narrow](#) (char\_type \_\_c, char \_\_dfault) const
- virtual const char\_type \* [do\\_narrow](#) (const char\_type \*\_\_lo, const char\_type \*\_\_hi, char \_\_dfault, char \*\_\_to) const
- virtual char\_type [do\\_tolower](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_tolower](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_toupper](#) (char\_type \_\_c) const
- virtual const char\_type \* [do\\_toupper](#) (char\_type \*\_\_lo, const char\_type \*\_\_hi) const
- virtual char\_type [do\\_widen](#) (char \_\_c) const
- virtual const char \* [do\\_widen](#) (const char \*\_\_lo, const char \*\_\_hi, char\_type \*\_\_to) const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- `__c_locale` `_M_c_locale_ctype`
- `bool` `_M_del`
- `char` `_M_narrow` `[1+static_cast< unsigned char >(-1)]`
- `char` `_M_narrow_ok`
- `const mask *` `_M_table`
- `__to_type` `_M_tolower`
- `__to_type` `_M_toupper`
- `char` `_M_widen` `[1+static_cast< unsigned char >(-1)]`
- `char` `_M_widen_ok`

## 4.427.1 Detailed Description

```
template<>
```

```
class std::ctype_byname< char >
```

22.2.1.4 Class `ctype_byname` specializations.

Definition at line 1499 of file `locale_facets.h`.

## 4.427.2 Member Typedef Documentation

4.427.2.1 `char_type`

```
typedef char std::ctype< char >::char_type [inherited]
```

Typedef for the template parameter `char`.

Definition at line 686 of file `locale_facets.h`.

## 4.427.3 Member Function Documentation

4.427.3.1 `classic_table()`

```
static const mask* std::ctype< char >::classic_table () throw () [static], [inherited]
```

Returns a pointer to the C locale mask table.

4.427.3.2 `do_narrow()` [1/2]

```
virtual char std::ctype< char >::do_narrow (
 char_type __c,
 char __dfault) const [inline], [protected], [virtual], [inherited]
```

Narrow `char`.

This virtual function converts the `char` to `char` using the simplest reasonable transformation. If the conversion fails, `dfault` is returned instead. For an underived `ctype<char>` facet, `c` will be returned unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char to convert.                |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted char.

Definition at line 1134 of file `locale_facets.h`.

**4.427.3.3 do\_narrow()** [2/2]

```
virtual const char_type* std::ctype< char >::do_narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *default* is used instead. For an undervied `ctype<char>` facet, the argument will be copied unchanged.

`do_narrow()` is a hook for a derived facet to change the behavior of narrowing. `do_narrow()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>__lo</code>      | Pointer to start of range.        |
| <code>__hi</code>      | Pointer to end of range.          |
| <code>__default</code> | Char to use if conversion fails.  |
| <code>__to</code>      | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 1160 of file `locale_facets.h`.

## 4.427.3.4 do\_tolower() [1/2]


```
virtual char_type std::ctype< char >::do_tolower (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                                                                                          |                      |
|------------------------------------------------------------------------------------------|----------------------|
| <br>__c | The char to convert. |
|------------------------------------------------------------------------------------------|----------------------|

## Returns

The lowercase char if convertible, else \_\_c.

## 4.427.3.5 do\_tolower() [2/2]



```
virtual const char_type* std::ctype< char >::do_tolower (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do\_tolower() is a hook for a derived facet to change the behavior of lowercasing. do\_tolower() must always return the same result for the same input.

## Parameters

|                                                                                             |                                 |
|---------------------------------------------------------------------------------------------|---------------------------------|
| <br>__lo | Pointer to first char in range. |
| <br>__hi | Pointer to end of range.        |

## Returns

\_\_hi.

**4.427.3.6 do\_toupper()** [1/2]


```
virtual char_type std::ctype< char >::do_toupper (
 char_type __c) const [protected], [virtual], [inherited]
```

Convert to uppercase.

This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                          |                      |
|------------------------------------------------------------------------------------------|----------------------|
| <br>__c | The char to convert. |
|------------------------------------------------------------------------------------------|----------------------|

**Returns**

The uppercase char if convertible, else \_\_c.

**4.427.3.7 do\_toupper()** [2/2]



```
virtual const char_type* std::ctype< char >::do_toupper (
 char_type * __lo,
 const char_type * __hi) const [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.

do\_toupper() is a hook for a derived facet to change the behavior of uppercasing. do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                             |                            |
|---------------------------------------------------------------------------------------------|----------------------------|
| <br>__lo | Pointer to start of range. |
| <br>__hi | Pointer to end of range.   |

**Returns**

\_\_hi.

4.427.3.8 do\_widen() [1/2]

```
virtual char_type std::ctype< char >::do_widen (
 char __c) const [inline], [protected], [virtual], [inherited]
```

Widen char.

This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

|                                       |                      |
|---------------------------------------|----------------------|
| <div><div>↔</div><div>__c</div></div> | The char to convert. |
|---------------------------------------|----------------------|

Returns

The converted character.

Definition at line 1084 of file locale\_facets.h.

4.427.3.9 do\_widen() [2/2]

```
virtual const char* std::ctype< char >::do_widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [protected], [virtual], [inherited]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do\_widen() is a hook for a derived facet to change the behavior of widening. do\_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

|                                        |                                   |
|----------------------------------------|-----------------------------------|
| <div><div>↔</div><div>__lo</div></div> | Pointer to start of range.        |
| <div><div>↔</div><div>__hi</div></div> | Pointer to end of range.          |
| <div><div>↔</div><div>__to</div></div> | Pointer to the destination array. |
| Generated by Doxygen                   |                                   |

**Returns**

`__hi`.

Definition at line 1107 of file `locale_facets.h`.

**4.427.3.10 is()** [1/2]

```
bool std::ctype< char >::is (
 mask __m,
 char __c) const [inline], [inherited]
```

Test char classification.

This function compares the mask `table[c]` to `__m`.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__c</code> | The char to compare the mask of. |
| <code>__m</code> | The mask to compare against.     |

**Returns**

True if `__m & table[__c]` is true, false otherwise.

Definition at line 43 of file `ctype_inline.h`.

**4.427.3.11 is()** [2/2]

```
const char * std::ctype< char >::is (
 const char * __lo,
 const char * __hi,
 mask * __vec) const [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char in the range `[lo, hi)` and successively writes it to `vec`. `vec` must have as many elements as the char array.

**Parameters**

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__lo</code>  | Pointer to start of range.           |
| <code>__hi</code>  | Pointer to end of range.             |
| <code>__vec</code> | Pointer to an array of mask storage. |



**Returns**

`__hi`.

Definition at line 48 of file ctype\_inline.h.

**4.427.3.12 narrow()** [1/2]

```
char std::ctype< char >::narrow (
 char_type __c,
 char __default) const [inline], [inherited]
```

Narrow char.

This function converts the char to char using the simplest reasonable transformation. If the conversion fails, default is returned instead. For an underived ctype<char> facet, c will be returned unchanged.

This function works as if it returns ctype<char>::do\_narrow(c). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

**Parameters**

|                        |                                     |
|------------------------|-------------------------------------|
| <code>__c</code>       | The char to convert.                |
| <code>__default</code> | Char to return if conversion fails. |

**Returns**

The converted character.

Definition at line 931 of file locale\_facets.h.

References std::ctype< \_CharT >::do\_narrow().

**4.427.3.13 narrow()** [2/2]

```
const char_type* std::ctype< char >::narrow (
 const char_type * __lo,
 const char_type * __hi,
 char __default,
 char * __to) const [inline], [inherited]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do\_narrow(lo, hi, dfault, to). do\_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

## Parameters

|                       |                                   |
|-----------------------|-----------------------------------|
| <code>__lo</code>     | Pointer to start of range.        |
| <code>__hi</code>     | Pointer to end of range.          |
| <code>__dfault</code> | Char to use if conversion fails.  |
| <code>__to</code>     | Pointer to the destination array. |

## Returns

`__hi`.

Definition at line 964 of file locale\_facets.h.

References `std::ctype< _CharT >::do_narrow()`.

## 4.427.3.14 scan\_is()

```
const char * std::ctype< char >::scan_is (
 mask __m,
 const char * __lo,
 const char * __hi) const [inline], [inherited]
```

Find char matching a mask.

This function searches for and returns the first char in [lo,hi) for which `is(m,char)` is true.

## Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>↵<br/>__m</code>  | The mask to compare against. |
| <code>↵<br/>__lo</code> | Pointer to start of range.   |
| <code>↵<br/>__hi</code> | Pointer to end of range.     |

## Returns

Pointer to a matching char if found, else `__hi`.

Definition at line 57 of file ctype\_inline.h.

#### 4.427.3.15 scan\_not()

```
const char * std::ctype< char >::scan_not (
 mask __m,
 const char * __lo,
 const char * __hi) const [inline], [inherited]
```

Find char not matching a mask.

This function searches for and returns a pointer to the first char in [\_\_lo,\_\_hi) for which is(m,char) is false.

##### Parameters

|                         |                              |
|-------------------------|------------------------------|
| <code>↵<br/>__m</code>  | The mask to compare against. |
| <code>↵<br/>__lo</code> | Pointer to start of range.   |
| <code>↵<br/>__hi</code> | Pointer to end of range.     |

##### Returns

Pointer to a non-matching char if found, else `__hi`.

Definition at line 67 of file ctype\_inline.h.

#### 4.427.3.16 table()

```
const mask* std::ctype< char >::table () const throw () [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic\_table() if none was provided.

Definition at line 983 of file locale\_facets.h.

#### 4.427.3.17 tolower() [1/2]


```
char_type std::ctype< char >::tolower (
 char_type __c) const [inline], [inherited]
```

Convert to lowercase.

This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

tolower() acts as if it returns ctype<char>::do\_tolower(\_\_c). do\_tolower() must always return the same result for the same input.

## Parameters

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | The char to convert. |
| <code>__c</code>                                                                  |                      |

## Returns

The lowercase char if convertible, else `__c`.

Definition at line 835 of file locale\_facets.h.

References `std::ctype< _CharT >::do_tolower()`.

## 4.427.3.18 tolower() [2/2]



```
const char_type* std::ctype< char >::tolower (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

`tolower()` acts as if it returns `ctype<char>::do_tolower(__lo, __hi)`. `do_tolower()` must always return the same result for the same input.

## Parameters

|                                                                                     |                                 |
|-------------------------------------------------------------------------------------|---------------------------------|
|  | Pointer to first char in range. |
| <code>__lo</code>                                                                   |                                 |
|  | Pointer to end of range.        |
| <code>__hi</code>                                                                   |                                 |

## Returns

`__hi`.

Definition at line 852 of file locale\_facets.h.

References `std::ctype< _CharT >::do_tolower()`.

**4.427.3.19 toupper()** [1/2]


```
char_type std::ctype< char >::toupper (
 char_type __c) const [inline], [inherited]
```

Convert to uppercase.

This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.

toupper() acts as if it returns ctype<char>::do\_toupper(c). do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                    |                      |
|----------------------------------------------------------------------------------------------------|----------------------|
|  <code>__c</code> | The char to convert. |
|----------------------------------------------------------------------------------------------------|----------------------|

**Returns**

The uppercase char if convertible, else `__c`.

Definition at line 802 of file locale\_facets.h.

References std::ctype<\_CharT>::do\_toupper().

**4.427.3.20 toupper()** [2/2]



```
const char_type* std::ctype< char >::toupper (
 char_type * __lo,
 const char_type * __hi) const [inline], [inherited]
```

Convert array to uppercase.

This function converts each char in the range [`__lo`,`__hi`) to uppercase if possible. Other chars remain untouched.

toupper() acts as if it returns ctype<char>::do\_toupper(`__lo`, `__hi`). do\_toupper() must always return the same result for the same input.

**Parameters**

|                                                                                                       |                                 |
|-------------------------------------------------------------------------------------------------------|---------------------------------|
|  <code>__lo</code> | Pointer to first char in range. |
|  <code>__hi</code> | Pointer to end of range.        |

**Returns**`__hi.`

Definition at line 819 of file locale\_facets.h.

References `std::ctype< _CharT >::do_toupper()`.

**4.427.3.21 widen() [1/2]**

```
char_type std::ctype< char >::widen (
 char __c) const [inline], [inherited]
```

Widen char.

This function converts the char to char\_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__c</code> | The char to convert. |
|------------------|----------------------|

**Returns**

The converted character.

Definition at line 872 of file locale\_facets.h.

References `std::ctype< _CharT >::do_widen()`.

**4.427.3.22 widen() [2/2]**

```
const char* std::ctype< char >::widen (
 const char * __lo,
 const char * __hi,
 char_type * __to) const [inline], [inherited]
```

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns `ctype<char>::do_widen(c)`. `do_widen()` must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See `codecvt` for that.

**Parameters**

|                        |                                   |
|------------------------|-----------------------------------|
| <code>↵<br/>_lo</code> | Pointer to first char in range.   |
| <code>↵<br/>_hi</code> | Pointer to end of range.          |
| <code>↵<br/>_to</code> | Pointer to the destination array. |

**Returns**

`__hi`.

Definition at line 899 of file `locale_facets.h`.

References `std::ctype<_CharT>::do_widen()`.

**4.427.4 Member Data Documentation****4.427.4.1 id**

```
locale::id std::ctype< char >::id [static], [inherited]
```

The facet id for `ctype<char>`

Definition at line 703 of file `locale_facets.h`.

**4.427.4.2 table\_size**

```
const size_t std::ctype< char >::table_size [static], [inherited]
```

The size of the mask table. It is `SCHAR_MAX + 1`.

Definition at line 705 of file `locale_facets.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)



4.428 `__gnu_cxx::debug_allocator<_Alloc>` Class Template Reference

## Public Types

- typedef `_Traits::const_pointer` **const\_pointer**
- typedef `_Traits::const_reference` **const\_reference**
- typedef `_Traits::difference_type` **difference\_type**
- typedef `_Traits::pointer` **pointer**
- typedef `_Traits::reference` **reference**
- typedef `_Traits::size_type` **size\_type**
- typedef `_Traits::value_type` **value\_type**

## Public Member Functions

- template<typename `_Alloc2`>  
**debug\_allocator** (const [debug\\_allocator](#)< `_Alloc2`> &\_\_a2, typename `__convertible<_Alloc2>::__type=0`)
- **debug\_allocator** (const `_Alloc` &\_\_a)
- pointer **allocate** (size\_type \_\_n)
- pointer **allocate** (size\_type \_\_n, const void \* \_\_hint)
- void **construct** (pointer \_\_p, const value\_type &\_\_val)
- template<typename `_Tp`, typename... `_Args`>  
void **construct** (`_Tp` \* \_\_p, `_Args` &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename `_Tp`>  
void **destroy** (`_Tp` \* \_\_p)
- size\_type **max\_size** () const throw ()

## Friends

- template<typename >  
class **debug\_allocator**
- template<typename `_Alloc2`>  
bool **operator!=** (const [debug\\_allocator](#) &\_\_lhs, const [debug\\_allocator](#)< `_Alloc2`> &\_\_rhs) noexcept
- template<typename `_Alloc2`>  
bool **operator==** (const [debug\\_allocator](#) &\_\_lhs, const [debug\\_allocator](#)< `_Alloc2`> &\_\_rhs) noexcept

## 4.428.1 Detailed Description

```
template<typename _Alloc>
class __gnu_cxx::debug_allocator< _Alloc>
```

A meta-allocator with debugging bits.

This is precisely the allocator defined in the C++03 Standard.

Definition at line 60 of file `debug_allocator.h`.

The documentation for this class was generated from the following file:

- [debug\\_allocator.h](#)

## 4.429 `std::decay<_Tp>` Class Template Reference

### Public Types

- `typedef __decay_selector< __remove_type >::__type type`

### 4.429.1 Detailed Description

```
template<typename _Tp>
class std::decay<_Tp>
```

`decay`

Definition at line 2147 of file `type_traits`.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 4.430 `std::decimal::decimal128` Class Reference

### Public Types

- `typedef float __decfloat128`

### Public Member Functions

- `decimal128 (decimal32 d32)`
- `decimal128 (decimal64 d64)`
- `decimal128 (float __r)`
- `decimal128 (double __r)`
- `decimal128 (long double __r)`
- `decimal128 (int __z)`
- `decimal128 (unsigned int __z)`
- `decimal128 (long __z)`
- `decimal128 (unsigned long __z)`
- `decimal128 (long long __z)`
- `decimal128 (unsigned long long __z)`
- `decimal128 (__decfloat128 __z)`
- `__decfloat128 __getval (void)`
- `void __setval (__decfloat128 __x)`
- `operator long long () const`
- `decimal128 & operator*= (decimal32 __rhs)`
- `decimal128 & operator*= (decimal64 __rhs)`
- `decimal128 & operator*= (decimal128 __rhs)`
- `decimal128 & operator*= (int __rhs)`

- [decimal128](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator\*=** (long \_\_rhs)
- [decimal128](#) & **operator\*=** (long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator++** ()
- [decimal128](#) **operator++** (int)
- [decimal128](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator+=** (long \_\_rhs)
- [decimal128](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator+=** (long long \_\_rhs)
- [decimal128](#) & **operator+=** (int \_\_rhs)
- [decimal128](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator--** ()
- [decimal128](#) **operator--** (int)
- [decimal128](#) & **operator-=** (int \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator-=** (long long \_\_rhs)
- [decimal128](#) & **operator-=** (long \_\_rhs)
- [decimal128](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal128](#) & **operator/=** (long \_\_rhs)
- [decimal128](#) & **operator/=** (long long \_\_rhs)
- [decimal128](#) & **operator/=** (int \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal128](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal128](#) & **operator/=** ([decimal32](#) \_\_rhs)

#### 4.430.1 Detailed Description

##### 3.2.4 Class decimal128.

Definition at line 399 of file decimal.

#### 4.430.2 Constructor & Destructor Documentation

#### 4.430.2.1 decimal128()

```
std::decimal::decimal128::decimal128 (
 __decfloat128 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 424 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

### 4.431 std::decimal::decimal32 Class Reference

#### Public Types

- typedef float **\_\_decfloat32**

#### Public Member Functions

- **decimal32** ([decimal64](#) \_\_d64)
- **decimal32** ([decimal128](#) \_\_d128)
- **decimal32** (float \_\_r)
- **decimal32** (double \_\_r)
- **decimal32** (long double \_\_r)
- **decimal32** (int \_\_z)
- **decimal32** (unsigned int \_\_z)
- **decimal32** (long \_\_z)
- **decimal32** (unsigned long \_\_z)
- **decimal32** (long long \_\_z)
- **decimal32** (unsigned long long \_\_z)
- [decimal32](#) (\_\_decfloat32 \_\_z)
- \_\_decfloat32 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat32 \_\_x)
- **operator long long** () const
- [decimal32](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator\*=** (int \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator\*=** (long \_\_rhs)
- [decimal32](#) & **operator\*=** (long long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator++** ()
- [decimal32](#) **operator++** (int)
- [decimal32](#) & **operator+=** ([decimal32](#) \_\_rhs)

- [decimal32](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator+=** (long \_\_rhs)
- [decimal32](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator+=** (long long \_\_rhs)
- [decimal32](#) & **operator+=** (int \_\_rhs)
- [decimal32](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator--** ()
- [decimal32](#) **operator--** (int)
- [decimal32](#) & **operator-=** (int \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator-=** (long long \_\_rhs)
- [decimal32](#) & **operator-=** (long \_\_rhs)
- [decimal32](#) & **operator-=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator-=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal64](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned long long \_\_rhs)
- [decimal32](#) & **operator/=** (long \_\_rhs)
- [decimal32](#) & **operator/=** (long long \_\_rhs)
- [decimal32](#) & **operator/=** (int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal128](#) \_\_rhs)
- [decimal32](#) & **operator/=** (unsigned int \_\_rhs)
- [decimal32](#) & **operator/=** ([decimal32](#) \_\_rhs)

#### 4.431.1 Detailed Description

##### 3.2.2 Class decimal32.

Definition at line 227 of file decimal.

#### 4.431.2 Constructor & Destructor Documentation

##### 4.431.2.1 decimal32()

```
std::decimal::decimal32::decimal32 (
 __decfloat32 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 251 of file decimal.

The documentation for this class was generated from the following file:

- [decimal](#)

## 4.432 std::decimal::decimal64 Class Reference

### Public Types

- typedef float **\_\_decfloat64**

### Public Member Functions

- **decimal64** ([decimal32](#) d32)
- **decimal64** ([decimal128](#) d128)
- **decimal64** (float \_\_r)
- **decimal64** (double \_\_r)
- **decimal64** (long double \_\_r)
- **decimal64** (int \_\_z)
- **decimal64** (unsigned int \_\_z)
- **decimal64** (long \_\_z)
- **decimal64** (unsigned long \_\_z)
- **decimal64** (long long \_\_z)
- **decimal64** (unsigned long long \_\_z)
- [decimal64](#) (\_\_decfloat64 \_\_z)
- \_\_decfloat64 **\_\_getval** (void)
- void **\_\_setval** (\_\_decfloat64 \_\_x)
- **operator long long** () const
- [decimal64](#) & **operator\*=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator\*=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator\*=** (int \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator\*=** (long \_\_rhs)
- [decimal64](#) & **operator\*=** (long long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator\*=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator++** ()
- [decimal64](#) **operator++** (int)
- [decimal64](#) & **operator+=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned int \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long long \_\_rhs)
- [decimal64](#) & **operator+=** (long \_\_rhs)
- [decimal64](#) & **operator+=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator+=** (long long \_\_rhs)
- [decimal64](#) & **operator+=** (int \_\_rhs)
- [decimal64](#) & **operator+=** ([decimal128](#) \_\_rhs)
- [decimal64](#) & **operator--** ()
- [decimal64](#) **operator--** (int)
- [decimal64](#) & **operator-=** (int \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal32](#) \_\_rhs)
- [decimal64](#) & **operator-=** (unsigned long \_\_rhs)
- [decimal64](#) & **operator-=** ([decimal64](#) \_\_rhs)
- [decimal64](#) & **operator-=** (long long \_\_rhs)

- `decimal64` & `operator=` (long `__rhs`)
- `decimal64` & `operator=` (`decimal128` `__rhs`)
- `decimal64` & `operator=` (unsigned int `__rhs`)
- `decimal64` & `operator=` (unsigned long long `__rhs`)
- `decimal64` & `operator/=` (`decimal64` `__rhs`)
- `decimal64` & `operator/=` (unsigned long `__rhs`)
- `decimal64` & `operator/=` (unsigned long long `__rhs`)
- `decimal64` & `operator/=` (long `__rhs`)
- `decimal64` & `operator/=` (long long `__rhs`)
- `decimal64` & `operator/=` (int `__rhs`)
- `decimal64` & `operator/=` (`decimal128` `__rhs`)
- `decimal64` & `operator/=` (unsigned int `__rhs`)
- `decimal64` & `operator/=` (`decimal32` `__rhs`)

#### 4.432.1 Detailed Description

##### 3.2.3 Class `decimal64`.

Definition at line 313 of file `decimal`.

#### 4.432.2 Constructor & Destructor Documentation

##### 4.432.2.1 `decimal64()`

```
std::decimal::decimal64::decimal64 (
 __decfloat64 __z) [inline]
```

Conforming extension: Conversion from scalar decimal type.

Definition at line 337 of file `decimal`.

The documentation for this class was generated from the following file:

- `decimal`

#### 4.433 `__gnu_pbds::detail::default_comb_hash_fn` Struct Reference

##### Public Types

- typedef `direct_mask_range_hashing` type

#### 4.433.1 Detailed Description

Primary template, default\_comb\_hash\_fn.

Definition at line 80 of file standard\_policies.hpp.

#### 4.433.2 Member Typedef Documentation

##### 4.433.2.1 type

```
typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type
```

Dispatched type.

Definition at line 83 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

### 4.434 std::default\_delete< \_Tp > Struct Template Reference

#### Public Member Functions

- constexpr [default\\_delete](#) () noexcept=default
- template<typename \_Up , typename = \_Require<is\_convertible<\_Up\*, \_Tp\*>>>>  
[default\\_delete](#) (const [default\\_delete](#)< \_Up > &) noexcept
- void [operator\(\)](#) (\_Tp \*\_\_ptr) const

#### 4.434.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete< _Tp >
```

Primary template of default\_delete, used by unique\_ptr for single objects.

Definition at line 63 of file unique\_ptr.h.

#### 4.434.2 Constructor & Destructor Documentation



4.434.2.1 `default_delete()` [1/2]

```
template<typename _Tp >
constexpr std::default_delete<_Tp >::default_delete () [default], [noexcept]
```

Default constructor.

4.434.2.2 `default_delete()` [2/2]

```
template<typename _Tp >
template<typename _Up , typename = _Require<is_convertible<_Up*, _Tp*>>>
std::default_delete<_Tp >::default_delete (
 const default_delete<_Up > &) [inline], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for objects of another type, `_Up`, only if `_Up*` is convertible to `_Tp*`.

Definition at line 75 of file `unique_ptr.h`.

## 4.434.3 Member Function Documentation

4.434.3.1 `operator()()`

```
template<typename _Tp >
void std::default_delete<_Tp >::operator() (
 _Tp * __ptr) const [inline]
```

Calls `delete __ptr`

Definition at line 79 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

4.435 `std::default_delete<_Tp[]>` Struct Template Reference

## Public Member Functions

- constexpr `default_delete` ( ) noexcept=default
- template<typename `_Up` , typename = `_Require<is_convertible<_Up*[], _Tp*[]>>>`  
`default_delete` (const `default_delete<_Up[]> &`) noexcept
- template<typename `_Up` >  
`enable_if<is_convertible<_Up*[], _Tp*[]>::value >::type operator()` ( `_Up * __ptr` ) const

#### 4.435.1 Detailed Description

```
template<typename _Tp>
struct std::default_delete<_Tp[]>
```

Specialization of `default_delete` for arrays, used by `unique_ptr<T[]>`

Definition at line 94 of file `unique_ptr.h`.

#### 4.435.2 Constructor & Destructor Documentation

##### 4.435.2.1 `default_delete()` [1/2]

```
template<typename _Tp >
constexpr std::default_delete<_Tp[]>::default_delete () [default], [noexcept]
```

Default constructor.

##### 4.435.2.2 `default_delete()` [2/2]

```
template<typename _Tp >
template<typename _Up , typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
std::default_delete<_Tp[]>::default_delete (
 const default_delete<_Up[]> &) [inline], [noexcept]
```

Converting constructor.

Allows conversion from a deleter for arrays of another type, such as a const-qualified version of `_Tp`.

Conversions from types derived from `_Tp` are not allowed because it is undefined to `delete[]` an array of derived types through a pointer to the base type.

Definition at line 111 of file `unique_ptr.h`.

#### 4.435.3 Member Function Documentation

4.435.3.1 `operator()`

```
template<typename _Tp >
template<typename _Up >
enable_if<is_convertible<_Up(*)[], _Tp(*)[]>::value>::type std::default_delete< _Tp[]>::operator()
(
 _Up * __ptr) const [inline]
```

Calls `delete[] __ptr`

Definition at line 116 of file `unique_ptr.h`.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

4.436 `__gnu_pbds::detail::default_eq_fn< Key >` Struct Template Reference

## Public Types

- typedef `std::equal_to< Key > type`

## 4.436.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_eq_fn< Key >
```

Primary template, `default_eq_fn`.

Definition at line 67 of file `standard_policies.hpp`.

## 4.436.2 Member Typedef Documentation

4.436.2.1 `type`

```
template<typename Key>
typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type
```

Dispatched type.

Definition at line 70 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.437 `__gnu_pbds::detail::default_hash_fn< Key >` Struct Template Reference

##### Public Types

- `typedef std::tr1::hash< Key > type`

##### 4.437.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >
```

Primary template, `default_hash_fn`.

Definition at line 59 of file `standard_policies.hpp`.

##### 4.437.2 Member Typedef Documentation

###### 4.437.2.1 `type`

```
template<typename Key>
typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type
```

Dispatched type.

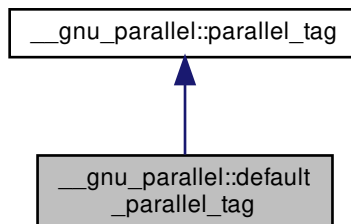
Definition at line 62 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.438 `__gnu_parallel::default_parallel_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::default_parallel_tag`:



## Public Member Functions

- `default_parallel_tag` (`_ThreadIndex` `__num_threads`)
- `_ThreadIndex` `__get_num_threads` ()
- void `set_num_threads` (`_ThreadIndex` `__num_threads`)

## 4.438.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

Definition at line 79 of file `tags.h`.

## 4.438.2 Member Function Documentation

4.438.2.1 `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file `tags.h`.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.438.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.439 `__gnu_pbds::detail::default_probe_fn`< `Comb_Probe_Fn` > Struct Template Reference

##### Public Types

- typedef `cond_type::__type` [type](#)

##### 4.439.1 Detailed Description

```
template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
```

Primary template, `default_probe_fn`.

Definition at line 117 of file `standard_policies.hpp`.

##### 4.439.2 Member Typedef Documentation

###### 4.439.2.1 `type`

```
template<typename Comb_Probe_Fn >
typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type
```

Dispatched type.

Definition at line 129 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.440 `__gnu_pbds::detail::default_resize_policy`< `Comb_Hash_Fn` > Struct Template Reference

##### Public Types

- typedef [hash\\_standard\\_resize\\_policy](#)< `size_policy_type`, [trigger](#), `false`, `size_type` > [type](#)

## 4.440.1 Detailed Description

```
template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
```

Primary template, `default_resize_policy`.

Definition at line 88 of file `standard_policies.hpp`.

## 4.440.2 Member Typedef Documentation

4.440.2.1 `type`

```
template<typename Comb_Hash_Fn>
typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_res
Comb_Hash_Fn >::type
```

Dispatched type.

Definition at line 105 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

4.441 `__gnu_pbds::detail::default_trie_access_traits< Key >` Struct Template Reference

## 4.441.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >
```

Primary template, `default_trie_access_traits`.

Definition at line 135 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.442 `__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >` Struct Template Reference

##### Public Types

- typedef [trie\\_string\\_access\\_traits< string\\_type > type](#)

##### 4.442.1 Detailed Description

```
template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >
```

Partial specialization, `default_trie_access_traits`.

Definition at line 142 of file `standard_policies.hpp`.

##### 4.442.2 Member Typedef Documentation

###### 4.442.2.1 `type`

```
template<typename Char , typename Char_Traits >
typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits<
std::basic_string< Char, Char_Traits, std::allocator< char > > >::type
```

Dispatched type.

Definition at line 149 of file `standard_policies.hpp`.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

#### 4.443 `__gnu_pbds::detail::default_update_policy` Struct Reference

##### Public Types

- typedef [lu\\_move\\_to\\_front\\_policy](#) `type`

##### 4.443.1 Detailed Description

Default update policy.

Definition at line 109 of file `standard_policies.hpp`.



## 4.443.2 Member Typedef Documentation

## 4.443.2.1 type

```
typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type
```

Dispatched type.

Definition at line 112 of file standard\_policies.hpp.

The documentation for this struct was generated from the following file:

- [standard\\_policies.hpp](#)

## 4.444 std::defer\_lock\_t Struct Reference

## 4.444.1 Detailed Description

Do not acquire ownership of the mutex.

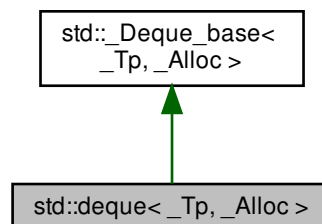
Definition at line 129 of file std\_mutex.h.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 4.445 std::deque&lt;\_Tp, \_Alloc&gt; Class Template Reference

Inheritance diagram for std::deque<\_Tp, \_Alloc>:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Base::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Base::iterator` **iterator**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `deque` ()=default
- `deque` (const allocator\_type &\_\_a)
- `deque` (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- `deque` (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- `deque` (const deque &\_\_x)
- `deque` (deque &&)=default
- `deque` (const deque &\_\_x, const allocator\_type &\_\_a)
- `deque` (deque &&\_\_x, const allocator\_type &\_\_a)
- `deque` (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  `deque` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- `~deque` ()
- void `assign` (size\_type \_\_n, const value\_type &\_\_val)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
  void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (initializer\_list< value\_type > \_\_l)
- reference `at` (size\_type \_\_n)
- const\_reference `at` (size\_type \_\_n) const
- reference `back` () noexcept
- const\_reference `back` () const noexcept
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- template<typename... \_Args>  
  `iterator emplace` (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
  void `emplace_back` (\_Args &&... \_\_args)
- template<typename... \_Args>  
  void `emplace_front` (\_Args &&... \_\_args)

- bool `empty` () const noexcept
- iterator `end` () noexcept
- const\_iterator `end` () const noexcept
- iterator `erase` (const\_iterator \_\_position)
- iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- reference `front` () noexcept
- const\_reference `front` () const noexcept
- allocator\_type `get_allocator` () const noexcept
- iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
- iterator `insert` (const\_iterator \_\_p, initializer\_list< value\_type > \_\_l)
- iterator `insert` (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator `insert` (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type `max_size` () const noexcept
- deque & `operator=` (const deque &\_\_x)
- deque & `operator=` (deque &&\_\_x) noexcept(\_Alloc\_traits::\_S\_always\_equal())
- deque & `operator=` (initializer\_list< value\_type > \_\_l)
- reference `operator[]` (size\_type \_\_n) noexcept
- const\_reference `operator[]` (size\_type \_\_n) const noexcept
- void `pop_back` () noexcept
- void `pop_front` () noexcept
- void `push_back` (const value\_type &\_\_x)
- void `push_back` (value\_type &&\_\_x)
- void `push_front` (const value\_type &\_\_x)
- void `push_front` (value\_type &&\_\_x)
- reverse\_iterator `rbegin` () noexcept
- const\_reverse\_iterator `rbegin` () const noexcept
- reverse\_iterator `rend` () noexcept
- const\_reverse\_iterator `rend` () const noexcept
- void `resize` (size\_type \_\_new\_size)
- void `resize` (size\_type \_\_new\_size, const value\_type &\_\_x)
- void `shrink_to_fit` () noexcept
- size\_type `size` () const noexcept
- void `swap` (deque &\_\_x) noexcept

#### Protected Types

- enum { `_S_initial_map_size` }
- typedef `__gnu_cxx::__alloc_traits`< \_Map\_alloc\_type > `_Map_alloc_traits`
- typedef \_Alloc\_traits::template rebind< \_Ptr >::other `_Map_alloc_type`
- typedef \_Alloc\_traits::pointer `_Ptr`
- typedef \_Alloc\_traits::const\_pointer `_Ptr_const`

## Protected Member Functions

- `_Map_pointer _M_allocate_map (size_t __n)`
- `_Ptr _M_allocate_node ()`
- `template<typename _InputIterator >`  
`void _M_assign_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_assign_aux (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `void _M_create_nodes (_Map_pointer __nstart, _Map_pointer __nfinish)`
- `void _M_deallocate_map (_Map_pointer __p, size_t __n) noexcept`
- `void _M_deallocate_node (_Ptr __p) noexcept`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize ()`
- `template<typename _Alloc1 >`  
`void _M_destroy_data (iterator __first, iterator __last, const _Alloc1 &)`
- `void _M_destroy_data (iterator __first, iterator __last, const std::allocator<_Tp> &)`
- `void _M_destroy_data_aux (iterator __first, iterator __last)`
- `void _M_destroy_nodes (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept`
- `iterator _M_erase (iterator __pos)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_begin (iterator __pos)`
- `void _M_erase_at_end (iterator __pos)`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void \_M\_fill\_initialize (const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Map_alloc_type _M_get_map_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `void \_M\_initialize\_map (size_t)`
- `template<typename... _Args>`  
`iterator _M_insert_aux (iterator __pos, _Args &&... __args)`
- `void _M_insert_aux (iterator __pos, size_type __n, const value_type &__x)`
- `template<typename _ForwardIterator >`  
`void _M_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)`
- `void _M_move_assign1 (deque &&__x, true\_type) noexcept`
- `void _M_move_assign1 (deque &&__x, false\_type)`
- `void _M_move_assign2 (deque &&__x, true\_type)`
- `void _M_move_assign2 (deque &&__x, false\_type)`
- `void \_M\_range\_check (size_type __n) const`
- `template<typename _InputIterator >`  
`void _M_range_insert_aux (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_insert_aux (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename... _Args>`  
`void _M_replace_map (_Args &&... __args)`
- `bool _M_shrink_to_fit ()`
- `template<typename _InputIterator >`  
`void \_M\_range\_initialize (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`

- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename... _Args>`  
`void _M_push_back_aux (_Args &&... __args)`
- `template<typename... _Args>`  
`void _M_push_front_aux (_Args &&... __args)`
- `void _M_pop_back_aux ()`
- `void _M_pop_front_aux ()`
- `iterator _M_reserve_elements_at_front (size_type __n)`
- `iterator _M_reserve_elements_at_back (size_type __n)`
- `void _M_new_elements_at_front (size_type __new_elements)`
- `void _M_new_elements_at_back (size_type __new_elements)`
- `void _M_reserve_map_at_back (size_type __nodes_to_add=1)`
- `void _M_reserve_map_at_front (size_type __nodes_to_add=1)`
- `void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)`

#### Static Protected Member Functions

- `static size_t _S_check_init_len (size_t __n, const allocator_type &__a)`
- `static size_type _S_max_size (const _Tp_alloc_type &__a) noexcept`

#### 4.445.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque<_Tp, _Alloc>
```

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the

### optional sequence requirements.

In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.

Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp\*\* \_M\_map
- size\_t \_M\_map\_size
- iterator \_M\_start, \_M\_finish

map\_size is at least 8. map is an array of map\_size pointers-to-nodes. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)

A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.

Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.

Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == \*(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node\_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↔node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.↔first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map\_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map\_size).
- A pointer in the range [map, map + map\_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

Here's the magic: nothing in deque is **aware** of the discontinuous storage!

The memory setup and layout occurs in the parent, \_Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

Definition at line 764 of file stl\_deque.h.

## 4.445.2 Constructor &amp; Destructor Documentation

## 4.445.2.1 deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque () [default]
```

Creates a deque with no elements.

## 4.445.2.2 deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 const allocator_type & __a) [inline], [explicit]
```

Creates a deque with no elements.

## Parameters

|                          |                      |
|--------------------------|----------------------|
| $\leftrightarrow$<br>__a | An allocator object. |
|--------------------------|----------------------|

Definition at line 841 of file stl\_deque.h.

## 4.445.2.3 deque() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a deque with default constructed elements.

## Parameters

|                          |                                             |
|--------------------------|---------------------------------------------|
| $\leftrightarrow$<br>__n | The number of elements to initially create. |
| $\leftrightarrow$<br>__a | An allocator.                               |

This constructor fills the deque with *n* default constructed elements.

Definition at line 854 of file stl\_deque.h.

#### 4.445.2.4 deque() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline]
```

Creates a deque with copies of an exemplar element.

##### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the deque with `__n` copies of `__value`.

Definition at line 866 of file `stl_deque.h`.

#### 4.445.2.5 deque() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 const deque< _Tp, _Alloc > & __x) [inline]
```

Deque copy constructor.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 893 of file `stl_deque.h`.

#### 4.445.2.6 deque() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 deque< _Tp, _Alloc > &&) [default]
```



Deque move constructor.

The newly-created deque contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified deque.

#### 4.445.2.7 deque() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 const deque< _Tp, _Alloc > & __x,
 const allocator_type & __a) [inline]
```

Copy constructor with alternative allocator.

Definition at line 912 of file stl\_deque.h.

#### 4.445.2.8 deque() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 deque< _Tp, _Alloc > && __x,
 const allocator_type & __a) [inline]
```

Move constructor with alternative allocator.

Definition at line 919 of file stl\_deque.h.

#### 4.445.2.9 deque() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
 initializer_list< value_type > __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from an initializer list.

##### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
| <code>__a</code> | An allocator object. |

Create a deque consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 952 of file `stl_deque.h`.

#### 4.445.2.10 `deque()` [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::deque<_Tp, _Alloc>::deque (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a deque from a range.

##### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a deque consisting of copies of the elements from `[__first, __last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(__first, __last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 979 of file `stl_deque.h`.

#### 4.445.2.11 `~deque()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque<_Tp, _Alloc>::~~deque () [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1003 of file `stl_deque.h`.

### 4.445.3 Member Function Documentation

#### 4.445.3.1 `_M_fill_initialize()`

```
template<typename _Tp, typename _Alloc>
void deque::_M_fill_initialize (
 const value_type & __value) [protected]
```

Fills the deque with copies of `value`.

**Parameters**

|                      |                |
|----------------------|----------------|
| <code>__value</code> | Initial value. |
|----------------------|----------------|

**Returns**

Nothing.

**Precondition**

`_M_start` and `_M_finish` have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).

Definition at line 394 of file `deque.tcc`.

Referenced by `std::deque< _StateSeqT >::deque()`.

**4.445.3.2 `_M_initialize_map()`**

```
template<typename _Tp , typename _Alloc >
void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (
 size_t __num_elements) [protected], [inherited]
```

Layout storage.

**Parameters**

|                             |                                                        |
|-----------------------------|--------------------------------------------------------|
| <code>__num_elements</code> | The count of T's for which to allocate space at first. |
|-----------------------------|--------------------------------------------------------|

**Returns**

Nothing.

The initial underlying memory layout is a bit complicated...

Definition at line 615 of file `stl_deque.h`.

**4.445.3.3 `_M_new_elements_at_back()`**

```
template<typename _Tp , typename _Alloc >
void deque::_M_new_elements_at_back (
 size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 904 of file deque.tcc.

Referenced by `std::deque<_StateSeqT>::_M_reserve_elements_at_back()`.

#### 4.445.3.4 `_M_new_elements_at_front()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_new_elements_at_front (
 size_type __new_elements) [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 879 of file deque.tcc.

Referenced by `std::deque<_StateSeqT>::_M_reserve_elements_at_front()`.

#### 4.445.3.5 `_M_pop_back_aux()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_pop_back_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Definition at line 558 of file deque.tcc.

#### 4.445.3.6 `_M_pop_front_aux()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_pop_front_aux () [protected]
```

Helper functions for `push_*` and `pop_*`.

Definition at line 574 of file deque.tcc.

#### 4.445.3.7 `_M_push_back_aux()`

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
void deque::_M_push_back_aux (
 _Args &&... __args) [protected]
```

Helper functions for `push_*` and `pop_*`.

Definition at line 482 of file deque.tcc.

Referenced by `std::deque<_StateSeqT>::push_back()`.

## 4.445.3.8 \_M\_push\_front\_aux()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
void deque::_M_push_front_aux (
 _Args &&... __args) [protected]
```

Helper functions for push\_\* and pop\_\*.

Definition at line 521 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::push\_front().

## 4.445.3.9 \_M\_range\_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (
 size_type __n) const [inline], [protected]
```

Safety check used only from at().

Definition at line 1351 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::at().

## 4.445.3.10 \_M\_range\_initialize() [1/2]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator >
void deque::_M_range_initialize (
 _InputIterator __first,
 _InputIterator __last,
 std::input_iterator_tag) [protected]
```

Fills the deque with whatever is in [first,last).

## Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

## Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push\_back on each value from the iterator.

Definition at line 420 of file deque.tcc.

Referenced by `std::deque<_StateSeqT>::deque()`.

#### 4.445.3.11 `_M_range_initialize()` [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _ForwardIterator >
void deque::_M_range_initialize (
 _ForwardIterator __first,
 _ForwardIterator __last,
 std::forward_iterator_tag) [protected]
```

Fills the deque with whatever is in [first,last).

##### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

##### Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using `push_back` on each value from the iterator.

Definition at line 444 of file deque.tcc.

#### 4.445.3.12 `_M_reallocate_map()`

```
template<typename _Tp , typename _Alloc >
void deque::_M_reallocate_map (
 size_type __nodes_to_add,
 bool __add_at_front) [protected]
```

Memory-handling helpers for the major map.

Makes sure the `_M_map` has space for new nodes. Does not actually add the nodes. Can invalidate `_M_map` pointers. (And consequently, deque iterators.)

Definition at line 929 of file deque.tcc.

Referenced by `std::deque<_StateSeqT>::_M_reserve_map_at_back()`, and `std::deque<_StateSeqT>::_M_reserve_map_at_front()`.

**4.445.3.13 \_M\_reserve\_elements\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back (
 size_type __n) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 2097 of file stl\_deque.h.

**4.445.3.14 \_M\_reserve\_elements\_at\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front (
 size_type __n) [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

Definition at line 2087 of file stl\_deque.h.

**4.445.3.15 \_M\_reserve\_map\_at\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
 size_type __nodes_to_add = 1) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 2123 of file stl\_deque.h.

**4.445.3.16 \_M\_reserve\_map\_at\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
 size_type __nodes_to_add = 1) [inline], [protected]
```

Memory-handling helpers for the major map.

Makes sure the \_M\_map has space for new nodes. Does not actually add the nodes. Can invalidate \_M\_map pointers. (And consequently, deque iterators.)

Definition at line 2131 of file stl\_deque.h.

**4.445.3.17 assign()** [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
 size_type __n,
 const value_type & __val) [inline]
```

Assigns a given value to a deque.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a deque with  $n$  copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

Definition at line 1066 of file `std_deque.h`.

**4.445.3.18 assign()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::deque<_Tp, _Alloc>::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a deque.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a deque with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

Definition at line 1085 of file `std_deque.h`.

**4.445.3.19 assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc>::assign (
 initializer_list<value_type> __l) [inline]
```

Assigns an initializer list to a deque.



## Parameters

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a deque with copies of the elements in the initializer\_list \_\_l.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

Definition at line 1110 of file stl\_deque.h.

## 4.445.3.20 at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::at (
 size_type __n) [inline]
```

Provides access to the data contained in the deque.

## Parameters

|   |                                                             |
|---|-------------------------------------------------------------|
| ↩ | The index of the element for which data should be accessed. |
| n |                                                             |

## Returns

Read/write reference to data.

## Exceptions

|                   |                             |
|-------------------|-----------------------------|
| std::out_of_range | If __n is an invalid index. |
|-------------------|-----------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out\_of\_range if the check fails.

Definition at line 1373 of file stl\_deque.h.

## 4.445.3.21 at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the deque.

#### Parameters

|                        |                                                             |
|------------------------|-------------------------------------------------------------|
| <code>_↵<br/>_n</code> | The index of the element for which data should be accessed. |
|------------------------|-------------------------------------------------------------|

#### Returns

Read-only (constant) reference to data.

#### Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws `out_of_range` if the check fails.

Definition at line 1391 of file `stl_deque.h`.

#### 4.445.3.22 `back()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

Definition at line 1424 of file `stl_deque.h`.

#### 4.445.3.23 `back()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

Definition at line 1437 of file `stl_deque.h`.

**4.445.3.24 begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1125 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::clear(), std::deque< \_StateSeqT >::deque(), std::deque< \_StateSeqT >::insert(), std::operator<(), std::deque< \_StateSeqT >::operator=(), std::operator==( ), and std::deque< \_StateSeqT >::~~deque().

**4.445.3.25 begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1133 of file stl\_deque.h.

**4.445.3.26 cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Definition at line 1196 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::insert().

**4.445.3.27 cend()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1205 of file stl\_deque.h.

#### 4.445.3.28 clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque<_Tp, _Alloc >::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1790 of file `stl_deque.h`.

#### 4.445.3.29 crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1214 of file `stl_deque.h`.

#### 4.445.3.30 crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque<_Tp, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1223 of file `stl_deque.h`.

#### 4.445.3.31 emplace()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
deque<_Tp, _Alloc >::iterator deque::emplace (
 const_iterator __position,
 _Args &&... __args)
```

Inserts an object in deque before specified iterator.

##### Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the deque. |
| <code>__args</code>     | Arguments.                                    |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.

Definition at line 188 of file deque.tcc.

Referenced by std::deque< \_StateSeqT >::insert().

**4.445.3.32 empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque< _Tp, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the deque is empty. (Thus begin() would equal end().)

Definition at line 1308 of file stl\_deque.h.

**4.445.3.33 end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1142 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::deque(), std::operator<(), std::deque< \_StateSeqT >::operator=(), std::deque< \_StateSeqT >::operator==(), and std::deque< \_StateSeqT >::~~deque().

**4.445.3.34 end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Definition at line 1151 of file stl\_deque.h.

**4.445.3.35 erase()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
 const_iterator __position) [inline]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1730 of file `stl_deque.h`.

**4.445.3.36 erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [`__first`,`__last`) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1754 of file `stl_deque.h`.

**4.445.3.37 front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

Definition at line 1402 of file `stl_deque.h`.

**4.445.3.38** front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

Definition at line 1413 of file stl\_deque.h.

**4.445.3.39** get\_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque< _Tp, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 1116 of file stl\_deque.h.

Referenced by std::deque< \_StateSeqT >::operator=().

**4.445.3.40** insert() [1/5]

```
template<typename _Tp , typename _Alloc >
deque< _Tp, _Alloc >::iterator deque::insert (
 const_iterator __position,
 const value_type & __x)
```

Inserts given value into deque before specified iterator.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__x</code>        | Data to be inserted.             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.

Definition at line 212 of file deque.tcc.

**4.445.3.41** `insert()` [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts given rvalue into deque before specified iterator.

**Parameters**

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__x</code>        | Data to be inserted.             |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

Definition at line 1616 of file `stl_deque.h`.

**4.445.3.42** `insert()` [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __p,
 initializer_list< value_type > __l) [inline]
```

Inserts an initializer list into the deque.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__p</code> | An iterator into the deque. |
| <code>__l</code> | An initializer_list.        |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the initializer\_list `__l` into the deque before the location specified by `__p`. This is known as *list insert*.

Definition at line 1630 of file `stl_deque.h`.



## 4.445.3.43 insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x) [inline]
```

Inserts a number of copies of given data into the deque.

## Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the deque.   |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

## Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by `__position`.

Definition at line 1649 of file `stl_deque.h`.

## 4.445.3.44 insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque< _Tp, _Alloc >::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Inserts a range into the deque.

## Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__position</code> | A const_iterator into the deque. |
| <code>__first</code>    | An input iterator.               |
| <code>__last</code>     | An input iterator.               |

## Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the deque before the location specified by `__position`. This is known as *range insert*.

Definition at line 1685 of file `stl_deque.h`.

#### 4.445.3.45 `max_size()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque<_Tp, _Alloc>::max_size () const [inline], [noexcept]
```

Returns the `size()` of the largest possible deque.

Definition at line 1235 of file `stl_deque.h`.

#### 4.445.3.46 `operator=()` [1/3]

```
template<typename _Tp , typename _Alloc >
deque<_Tp, _Alloc> & deque::operator= (
 const deque<_Tp, _Alloc> & __x)
```

Deque assignment operator.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

All the elements of `x` are copied.

The newly-created deque uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 96 of file `deque.tcc`.

#### 4.445.3.47 `operator=()` [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque<_Tp, _Alloc>::operator= (
 deque<_Tp, _Alloc> && __x) [inline], [noexcept]
```

Deque move assignment operator.

##### Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__x</code> | A deque of identical element and allocator types. |
|------------------|---------------------------------------------------|

The contents of `__x` are moved into this deque (without copying, if the allocators permit it). `__x` is a valid, but unspecified deque.

Definition at line 1028 of file `std_deque.h`.

#### 4.445.3.48 operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque& std::deque< _Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Assigns an initializer list to a deque.

##### Parameters

|                  |                      |
|------------------|----------------------|
| <code>↩</code>   | An initializer_list. |
| <code>__↩</code> |                      |
| <code>↩</code>   |                      |
| <code>__↩</code> |                      |
| <code>l</code>   |                      |

This function fills a deque with copies of the elements in the initializer\_list `__l`.

Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

Definition at line 1047 of file `std_deque.h`.

#### 4.445.3.49 operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::operator[] (
 size_type __n) [inline], [noexcept]
```

Subscript access to the data contained in the deque.

##### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__↩</code> | The index of the element for which data should be accessed. |
| <code>__n</code> |                                                             |

##### Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1324 of file `stl_deque.h`.

#### 4.445.3.50 `operator[]()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::operator[] (
 size_type __n) const [inline], [noexcept]
```

Subscript access to the data contained in the deque.

##### Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>__n</code> | The index of the element for which data should be accessed. |
|------------------|-------------------------------------------------------------|

##### Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1342 of file `stl_deque.h`.

#### 4.445.3.51 `pop_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1552 of file `stl_deque.h`.

#### 4.445.3.52 `pop_front()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the deque by one.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 1529 of file `stl_deque.h`.

## 4.445.3.53 push\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
 const value_type & __x) [inline]
```

Add data to the end of the deque.

## Parameters

|       |                   |
|-------|-------------------|
| $\_x$ | Data to be added. |
|-------|-------------------|

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1493 of file stl\_deque.h.

## 4.445.3.54 push\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (
 const value_type & __x) [inline]
```

Add data to the front of the deque.

## Parameters

|       |                   |
|-------|-------------------|
| $\_x$ | Data to be added. |
|-------|-------------------|

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.

Definition at line 1456 of file stl\_deque.h.

## 4.445.3.55 rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1160 of file stl\_deque.h.

**4.445.3.56** `rbegin()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

Definition at line 1169 of file `stl_deque.h`.

**4.445.3.57** `rend()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1178 of file `stl_deque.h`.

**4.445.3.58** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

Definition at line 1187 of file `stl_deque.h`.

**4.445.3.59** `resize()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
 size_type __new_size) [inline]
```

Resizes the deque to the specified number of elements.

**Parameters**

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain. |
|-------------------------|----------------------------------------------|

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

Definition at line 1249 of file stl\_deque.h.

#### 4.445.3.60 resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
 size_type __new_size,
 const value_type & __x) [inline]
```

Resizes the deque to the specified number of elements.

##### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the deque should contain.      |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

Definition at line 1271 of file stl\_deque.h.

#### 4.445.3.61 shrink\_to\_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::shrink_to_fit () [inline], [noexcept]
```

A non-binding request to reduce memory use.

Definition at line 1299 of file stl\_deque.h.

#### 4.445.3.62 size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::size () const [inline], [noexcept]
```

Returns the number of elements in the deque.

Definition at line 1230 of file stl\_deque.h.

Referenced by `std::deque< _StateSeqT >::_M_range_check()`, `std::deque< _StateSeqT >::operator=()`, `std::deque< _StateSeqT >::operator==()`, and `std::deque< _StateSeqT >::resize()`.

#### 4.445.3.63 swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::swap (
 deque< _Tp, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another deque.

## Parameters

|                 |                                                  |
|-----------------|--------------------------------------------------|
| <code>_↔</code> | A deque of the same element and allocator types. |
| <code>_X</code> |                                                  |

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(d1,d2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

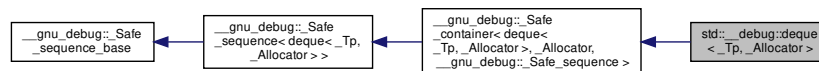
Definition at line 1772 of file `stl_deque.h`.

The documentation for this class was generated from the following files:

- [stl\\_deque.h](#)
- [deque.tcc](#)

#### 4.446 `std::__debug::deque<_Tp, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::deque<_Tp, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug__Safe_iterator<_Base_const_iterator, deque>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator<const_iterator>` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug__Safe_iterator<_Base_iterator, deque>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator<iterator>` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**



## Public Member Functions

- **deque** (const [deque](#) &)=default
- **deque** ([deque](#) &&)=default
- **deque** (const [deque](#) &\_\_d, const \_Allocator &\_\_a)
- **deque** ([deque](#) &&\_\_d, const \_Allocator &\_\_a)
- **deque** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **deque** (const \_Allocator &\_\_a)
- **deque** (size\_type \_\_n, const \_Allocator &\_\_a=\_Allocator())
- **deque** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
**deque** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **deque** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & **\_M\_base** () noexcept
- const [\\_Base](#) & **\_M\_base** () const noexcept
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_swap** (\_Safe\_container &\_\_x) noexcept
- void **\_M\_transfer\_from\_if** (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_t)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- reference **back** () noexcept
- const\_reference **back** () const noexcept
- [iterator](#) **begin** () noexcept
- const\_iterator **begin** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- const\_reverse\_iterator **crbegin** () const noexcept
- const\_reverse\_iterator **crend** () const noexcept
- template<typename... \_Args>  
[iterator](#) **emplace** (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&... \_\_args)
- template<typename... \_Args>  
void **emplace\_front** (\_Args &&... \_\_args)
- [iterator](#) **end** () noexcept
- const\_iterator **end** () const noexcept
- [iterator](#) **erase** (const\_iterator \_\_position)
- [iterator](#) **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- reference **front** () noexcept
- const\_reference **front** () const noexcept
- [iterator](#) **insert** (const\_iterator \_\_position, const \_Tp &\_\_x)
- [iterator](#) **insert** (const\_iterator \_\_position, \_Tp &&\_\_x)
- [iterator](#) **insert** (const\_iterator \_\_position, [initializer\\_list](#)< value\_type > \_\_l)
- [iterator](#) **insert** (const\_iterator \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator, typename = std::RequireInputIter<\_InputIterator>>>  
[iterator](#) **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- [deque](#) & **operator=** (const [deque](#) &)=default
- [deque](#) & **operator=** ([deque](#) &&)=default

- `deque` & `operator=` (`initializer_list`< `value_type` > `__l`)
- reference `operator[]` (`size_type __n`) noexcept
- const\_reference `operator[]` (`size_type __n`) const noexcept
- void `pop_back` () noexcept
- void `pop_front` () noexcept
- void `push_back` (const `_Tp` & `__x`)
- void `push_back` (`_Tp` && `__x`)
- void `push_front` (const `_Tp` & `__x`)
- void `push_front` (`_Tp` && `__x`)
- `reverse_iterator` `rbegin` () noexcept
- const `reverse_iterator` `rbegin` () const noexcept
- `reverse_iterator` `rend` () noexcept
- const `reverse_iterator` `rend` () const noexcept
- void `resize` (`size_type __sz`)
- void `resize` (`size_type __sz`, const `_Tp` & `__c`)
- void `shrink_to_fit` () noexcept
- void `swap` (`deque` & `__x`) noexcept(*/\*conditional \*/*)

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

#### Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_invalidate_all` () const
- void `_M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () noexcept
- void `_M_swap` (`_Safe_sequence_base` & `__x`) noexcept

#### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
`class ::__gnu_debug::_Safe_iterator`

#### 4.446.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::deque<_Tp, _Allocator>
```

Class `std::deque` with safety/checking/debug instrumentation.

Definition at line 36 of file `debug/deque`.

#### 4.446.2 Member Function Documentation

##### 4.446.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base()`.

##### 4.446.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, `i->_M_version == _M_version`.

##### 4.446.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence<map<_Key, _Tp, _Compare, _Allocator>>::_M_transfer_from_if()`.

##### 4.446.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.446.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 4.446.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.446.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.446.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< deque< _Tp, _Allocator > >::_M_transfer_from_if (
 _Safe_sequence< deque< _Tp, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 4.446.3 Member Data Documentation

#### 4.446.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

4.446.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator> >::_M_transfer_from_if()`.

4.446.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/deque](#)

4.447 `std::tr2::direct_bases<_Tp>` Struct Template Reference

## Public Types

- typedef [\\_\\_reflection\\_typelist<\\_\\_direct\\_bases\(\\_Tp\)...>](#) **type**

## 4.447.1 Detailed Description

```
template<typename _Tp>
struct std::tr2::direct_bases<_Tp>
```

Enumerate all the direct base classes of a class. Form of a typelist.

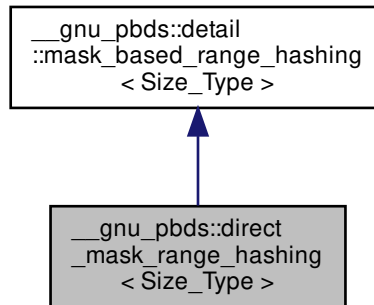
Definition at line 95 of file `tr2/type_traits`.

The documentation for this struct was generated from the following file:

- [tr2/type\\_traits](#)

#### 4.448 `__gnu_pbds::direct_mask_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mask_range_hashing< Size_Type >`:



##### Public Types

- typedef `Size_Type` **size\_type**

##### Public Member Functions

- void **swap** ([direct\\_mask\\_range\\_hashing< Size\\_Type >](#) &other)

##### Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type [operator\(\)](#) (size\_type hash) const
- size\_type **range\_hash** (size\_type hash) const
- void **swap** (mask\_based\_range\_hashing &other)

##### 4.448.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >
```

A mask range-hashing class (uses a bitmask).

Definition at line 109 of file `hash_policy.hpp`.

## 4.448.2 Member Function Documentation

4.448.2.1 `operator()`

```
template<typename Size_Type >
direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_Type >::operator() (
 size_type hash) const [inline], [protected]
```

Transforms the `__hash` value `hash` into a ranged-hash value (using a bit-mask).

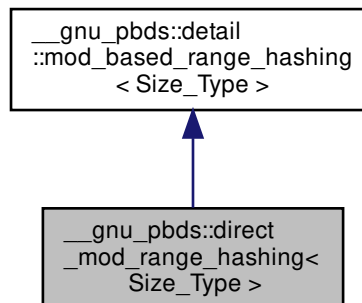
Definition at line 59 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

4.449 `__gnu_pbds::direct_mod_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::direct_mod_range_hashing< Size_Type >`:



## Public Types

- typedef `Size_Type` **size\_type**

## Public Member Functions

- void **swap** ([direct\\_mod\\_range\\_hashing< Size\\_Type >](#) &other)

### Protected Member Functions

- void **notify\_resized** (size\_type size)
- size\_type **operator()** (size\_type hash) const
- size\_type **range\_hash** (size\_type s) const
- void **swap** (mod\_based\_range\_hashing &other)

#### 4.449.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >
```

A mod range-hashing class (uses the modulo function).

Definition at line 141 of file hash\_policy.hpp.

#### 4.449.2 Member Function Documentation

##### 4.449.2.1 operator>()

```
template<typename Size_Type >
direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type
>::operator() (
 size_type hash) const [inline], [protected]
```

Transforms the \_\_hash value hash into a ranged-hash value (using a modulo operation).

Definition at line 59 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.450 std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r > Class Template Reference

### Public Types

- template<typename \_Sseq >  
using **\_If\_seed\_seq** = typename [enable\\_if](#)< \_\_detail::\_\_is\_seed\_seq< \_Sseq, [discard\\_block\\_engine](#),  
[result\\_type](#) >::value >::type
- typedef \_RandomNumberEngine::result\_type [result\\_type](#)



## Public Member Functions

- `discard_block_engine` ()
- `discard_block_engine` (const `_RandomNumberEngine` &\_\_rng)
- `discard_block_engine` (`_RandomNumberEngine` &&\_\_rng)
- `discard_block_engine` (`result_type` \_\_s)
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`discard_block_engine` (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & `base` () const noexcept
- void `discard` (unsigned long long \_\_z)
- `result_type operator`() ()
- void `seed` ()
- void `seed` (`result_type` \_\_s)
- `template<typename _Sseq >`  
`_If_seed_seq<_Sseq >` `seed` (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr `result_type` `max` ()
- static constexpr `result_type` `min` ()

## Static Public Attributes

- static constexpr size\_t `block_size`
- static constexpr size\_t `used_block`

## Friends

- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`  
`std::basic_ostream<_CharT, _Traits >` & `operator<<` (`std::basic_ostream<_CharT, _Traits >` &\_\_os, const `std::discard_block_engine<_RandomNumberEngine1, __p1, __r1 >` &\_\_x)
- bool `operator==` (const `discard_block_engine` &\_\_lhs, const `discard_block_engine` &\_\_rhs)
- `template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits >`  
`std::basic_istream<_CharT, _Traits >` & `operator>>` (`std::basic_istream<_CharT, _Traits >` &\_\_is, `std::discard_block_engine<_RandomNumberEngine1, __p1, __r1 >` &\_\_x)

## 4.450.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine<_RandomNumberEngine, __p, __r>
```

Produces random numbers from some base engine by discarding blocks of data.

`0 <= __r <= __p`

Definition at line 884 of file random.h.

#### 4.450.2 Member Typedef Documentation

##### 4.450.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

Definition at line 887 of file random.h.

#### 4.450.3 Constructor & Destructor Documentation

##### 4.450.3.1 discard\_block\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine () [inline]
```

Constructs a default discard\_block\_engine engine.

The underlying engine is default constructed as well.

Definition at line 906 of file random.h.

##### 4.450.3.2 discard\_block\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

##### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 916 of file random.h.

## 4.450.3.3 discard\_block\_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a discard\_block\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 926 of file random.h.

## 4.450.3.4 discard\_block\_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 result_type __s) [inline], [explicit]
```

Seed constructs a discard\_block\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

Definition at line 936 of file random.h.

## 4.450.3.5 discard\_block\_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq , typename = _If_seed_seq<_Sseq>>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
 _Sseq & __q) [inline], [explicit]
```

Generator construct a discard\_block\_engine engine.

**Parameters**

|                  |                  |
|------------------|------------------|
| <code>__q</code> | A seed sequence. |
|------------------|------------------|

Definition at line 946 of file random.h.

**4.450.4 Member Function Documentation****4.450.4.1 base()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine& std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ()
const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 990 of file random.h.

**4.450.4.2 discard()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 1011 of file random.h.

**4.450.4.3 max()**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::max ()
[inline], [static]
```

Gets the maximum value in the generated random number range.

Definition at line 1004 of file random.h.

References `std::max()`.

#### 4.450.4.4 min()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine<_RandomNumberEngine, __p, __r>::min ()
[inline], [static]
```

Gets the minimum value in the generated random number range.

Definition at line 997 of file random.h.

References std::min().

#### 4.450.4.5 operator()()

```
template<typename _RandomNumberEngine , size_t __p, size_t __r>
discard_block_engine<_RandomNumberEngine, __p, __r>::result_type std::discard_block_engine<_↵
RandomNumberEngine, __p, __r>::operator() ()
```

Gets the next value in the generated random number sequence.

Definition at line 682 of file bits/random.tcc.

#### 4.450.4.6 seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed () [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 955 of file random.h.

#### 4.450.4.7 seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
 result_type __s) [inline]
```

Reseeds the discard\_block\_engine object with the default seed for the underlying base class generator engine.

Definition at line 966 of file random.h.

#### 4.450.4.8 seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq>
_If_seed_seq<_Sseq> std::discard_block_engine<_RandomNumberEngine, __p, __r>::seed (
 _Sseq & __q) [inline]
```

Reseeds the discard\_block\_engine object with the given seed sequence.

**Parameters**

|                 |                            |
|-----------------|----------------------------|
| <code>_↵</code> | A seed generator function. |
| <code>_q</code> |                            |

Definition at line 979 of file random.h.

**4.450.5 Friends And Related Function Documentation****4.450.5.1 operator<<**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↵
Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x) [friend]
```

Inserts the current state of a discard\_block\_engine random number generator engine \_\_x into the output stream \_\_os.

**Parameters**

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A discard_block_engine random number generator engine. |

**Returns**

The output stream with the state of \_\_x inserted or in an error state.

**4.450.5.2 operator==**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
 const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
 const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs) [friend]
```

Compares two discard\_block\_engine random number generator objects of the same type for equality.

**Parameters**

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A discard_block_engine random number generator object.       |
| <code>__rhs</code> | Another discard_block_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 1035 of file random.h.

**4.450.5.3 `operator>>`**

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::discard_block_engine<_RandomNumberEngine1, __p1, __r1> & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine `__x` from the input stream `__is`.

**Parameters**

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__is</code> | An input stream.                                       |
| <code>__x</code>  | A discard_block_engine random number generator engine. |

**Returns**

The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**4.451 `std::discrete_distribution<_IntType>` Class Template Reference****Classes**

- struct [param\\_type](#)

**Public Types**

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- `template<typename _InputIterator >`  
**discrete\_distribution** (`_InputIterator __wbegin, _InputIterator __wend`)
- **discrete\_distribution** (`initializer_list< double > __wl`)
- `template<typename _Func >`  
**discrete\_distribution** (`size_t __nw, double __xmin, double __xmax, _Func __fw`)
- **discrete\_distribution** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
`void __generate` (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
`void __generate` (`result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
`result_type operator()` (`_UniformRandomNumberGenerator &__urng, const param_type &__p`)
- `param_type param` () const
- `void param` (`const param_type &__param`)
- `std::vector< double > probabilities` () const
- `void reset` ()

## Friends

- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<<` (`std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x`)
- `bool operator==` (`const discrete_distribution &__d1, const discrete_distribution &__d2`)
- `template<typename _IntType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>>` (`std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x`)

## 4.451.1 Detailed Description

```
template<typename _IntType = int>
class std::discrete_distribution< _IntType >
```

A `discrete_distribution` random number distribution.

The formula for the discrete probability mass function is

Definition at line 5277 of file `random.h`.



#### 4.451.2 Member Typedef Documentation

##### 4.451.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 5280 of file random.h.

#### 4.451.3 Member Function Documentation

##### 4.451.3.1 max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5402 of file random.h.

References std::vector<\_Tp, \_Alloc>::empty(), and std::vector<\_Tp, \_Alloc>::size().

##### 4.451.3.2 min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5395 of file random.h.

##### 4.451.3.3 operator>()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::discrete_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 5413 of file random.h.

#### 4.451.3.4 param() [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5380 of file random.h.

#### 4.451.3.5 param() [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5388 of file random.h.

#### 4.451.3.6 probabilities()

```
template<typename _IntType = int>
std::vector<double> std::discrete_distribution< _IntType >::probabilities () const [inline]
```

Returns the probabilities of the distribution.

Definition at line 5370 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

#### 4.451.3.7 reset()

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Definition at line 5363 of file random.h.

## 4.451.4 Friends And Related Function Documentation

## 4.451.4.1 operator&lt;&lt;

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::discrete_distribution< _IntType1 > & __x) [friend]
```

Inserts a discrete\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

|      |                                                     |
|------|-----------------------------------------------------|
| __os | An output stream.                                   |
| __x  | A discrete_distribution random number distribution. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 4.451.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
 const discrete_distribution< _IntType > & __d1,
 const discrete_distribution< _IntType > & __d2) [friend]
```

Return true if two discrete distributions have the same parameters.

Definition at line 5448 of file random.h.

## 4.451.4.3 operator&gt;&gt;

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::discrete_distribution< _IntType1 > & __x) [friend]
```

Extracts a discrete\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                         |                                                                      |
|-------------------------|----------------------------------------------------------------------|
| <code>_↔<br/>_is</code> | An input stream.                                                     |
| <code>_↔<br/>_x</code>  | A <code>discrete_distribution</code> random number generator engine. |

## Returns

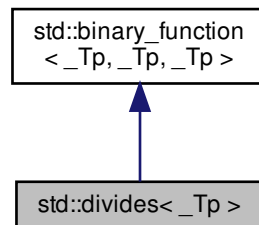
The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.452 `std::divides<_Tp>` Struct Template Reference

Inheritance diagram for `std::divides<_Tp>`:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &`__x`, const `_Tp` &`__y`) const

#### 4.452.1 Detailed Description

```
template<typename _Tp>
struct std::divides<_Tp>
```

One of the [math functors](#).

Definition at line 156 of file `stl_function.h`.

#### 4.452.2 Member Typedef Documentation

##### 4.452.2.1 `first_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.452.2.2 `result_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.452.2.3 `second_argument_type`

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.453 `std::divides< void >` Struct Template Reference

##### Public Types

- typedef `__is_transparent` **is\_transparent**

##### Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`←  
`t`)/`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)/`std::forward`< `_Up` >(`_u`))

##### 4.453.1 Detailed Description

```
template<>
struct std::divides< void >
```

One of the [math functors](#).

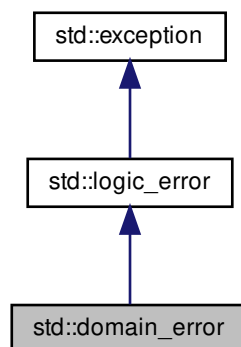
Definition at line 275 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.454 `std::domain_error` Class Reference

Inheritance diagram for `std::domain_error`:



#### Public Member Functions

- `domain_error` (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- `domain_error` (const char \*) \_GLIBCXX\_TXN\_SAFE
- `domain_error` (const [domain\\_error](#) &)=default
- `domain_error` ([domain\\_error](#) &&)=default
- `domain_error` & `operator=` (const [domain\\_error](#) &)=default
- `domain_error` & `operator=` ([domain\\_error](#) &&)=default
- virtual const char \* [what](#) () const noexcept

##### 4.454.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

Definition at line 153 of file `stdexcept`.

##### 4.454.2 Member Function Documentation

###### 4.454.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

#### 4.455 `__gnu_pbds::detail::dumnode_const_iterator`< Key, Data, \_Alloc > Struct Template Reference

#### Public Types

- typedef const\_iterator **const\_reference**
- typedef const\_reference **reference**
- typedef const\_iterator **value\_type**

#### 4.455.1 Detailed Description

```
template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >
```

Constant node iterator.

Definition at line 52 of file null\_node\_metadata.hpp.

The documentation for this struct was generated from the following file:

- [null\\_node\\_metadata.hpp](#)

#### 4.456 std::chrono::duration< \_Rep, \_Period > Struct Template Reference

##### Public Types

- using **period** = typename \_Period::type
- using **rep** = \_Rep

##### Public Member Functions

- **duration** (const [duration](#) &)=default
- template<typename \_Rep2 , typename = \_Require< is\_convertible<const \_Rep2&, rep>, \_\_or\_\_is\_float<rep>, \_\_not\_\_is\_↵ float<\_Rep2>>>>>
 constexpr **duration** (const \_Rep2 &\_\_rep)
- template<typename \_Rep2 , typename \_Period2 , typename = \_Require< is\_convertible<const \_Rep2&, rep>, \_\_or\_\_is\_float<rep>, \_\_and\_\_is\_harmonic<\_Period2>, \_\_not\_\_is\_float<\_Rep2>>>>>
 constexpr **duration** (const [duration](#)< \_Rep2, \_Period2 > &\_\_d)
- constexpr rep **count** () const
- template<typename \_Rep2 = rep>
 constexpr [enable\\_if](#)<![treat\\_as\\_floating\\_point](#)< \_Rep2 >::value, [duration](#) & >::type **operator**%= (const rep &\_\_rhs)
- template<typename \_Rep2 = rep>
 constexpr [enable\\_if](#)<![treat\\_as\\_floating\\_point](#)< \_Rep2 >::value, [duration](#) & >::type **operator**%= (const [duration](#) &\_\_d)
- constexpr [duration](#) & **operator**\*= (const rep &\_\_rhs)
- constexpr [duration](#)< typename [common\\_type](#)< rep >::type, period > **operator**+ () const
- constexpr [duration](#) & **operator**++ ()
- constexpr [duration](#) **operator**++ (int)
- constexpr [duration](#) & **operator**+= (const [duration](#) &\_\_d)
- constexpr [duration](#)< typename [common\\_type](#)< rep >::type, period > **operator**- () const
- constexpr [duration](#) & **operator**-- ()
- constexpr [duration](#) **operator**-- (int)
- constexpr [duration](#) & **operator**-= (const [duration](#) &\_\_d)
- constexpr [duration](#) & **operator**/= (const rep &\_\_rhs)
- [duration](#) & **operator**= (const [duration](#) &)=default



## Static Public Member Functions

- static constexpr `duration` **max** () noexcept
- static constexpr `duration` **min** () noexcept
- static constexpr `duration` **zero** () noexcept

## Related Functions

(Note that these are not member functions.)

- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
constexpr `common_type<duration<_Rep1, _Period1>, duration<_Rep2, _Period2>>`::type `operator+`  
(const `duration<_Rep1, _Period1>` &\_\_lhs, const `duration<_Rep2, _Period2>` &\_\_rhs)
- `template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2 >`  
constexpr `common_type<duration<_Rep1, _Period1>, duration<_Rep2, _Period2>>`::type `operator-` (const  
`duration<_Rep1, _Period1>` &\_\_lhs, const `duration<_Rep2, _Period2>` &\_\_rhs)
- `template<typename _Rep1, typename _Period, typename _Rep2 >`  
constexpr `duration<__common_rep_t<_Rep1, _Rep2>, _Period>` `operator*` (const `duration<_Rep1, _Period>` &\_\_d, const `_Rep2` &\_\_s)
- `template<typename _Rep1, typename _Rep2, typename _Period >`  
constexpr `duration<__common_rep_t<_Rep2, _Rep1>, _Period>` `operator*` (const `_Rep1` &\_\_s, const  
`duration<_Rep2, _Period>` &\_\_d)

## 4.456.1 Detailed Description

```
template<typename _Rep, typename _Period>
struct std::chrono::duration<_Rep, _Period>
```

`duration`

Definition at line 71 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

4.457 `std::chrono::duration_values<_Rep>` Struct Template Reference

## Static Public Member Functions

- static constexpr `_Rep` **max** () noexcept
- static constexpr `_Rep` **min** () noexcept
- static constexpr `_Rep` **zero** () noexcept

#### 4.457.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::duration_values< _Rep >
```

duration\_values

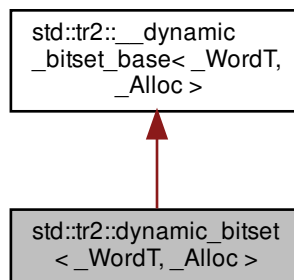
Definition at line 390 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### 4.458 std::tr2::dynamic\_bitset< \_WordT, \_Alloc > Class Template Reference

Inheritance diagram for std::tr2::dynamic\_bitset< \_WordT, \_Alloc >:



#### Classes

- class [reference](#)

#### Public Types

- typedef `__dynamic_bitset_base< _WordT, _Alloc >` **\_Base**
- typedef `_Alloc` **allocator\_type**
- typedef `_WordT` **block\_type**
- typedef `bool` **const\_reference**
- typedef `size_t` **size\_type**

## Public Member Functions

- [dynamic\\_bitset](#) ()=default
- [dynamic\\_bitset](#) (const allocator\_type &\_\_alloc)
- [dynamic\\_bitset](#) (size\_type \_\_nbits, unsigned long long \_\_val=0ULL, const allocator\_type &\_\_alloc=allocator\_type(), type())
- **dynamic\_bitset** (initializer\_list< block\_type > \_\_il, const allocator\_type &\_\_alloc=allocator\_type())
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
[dynamic\\_bitset](#) (const [std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_pos=0, typename [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::size\_type \_\_n=[std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 >::npos, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'), const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (const char \*\_\_str, const allocator\_type &\_\_alloc=allocator\_type())
- [dynamic\\_bitset](#) (const [dynamic\\_bitset](#) &)=default
- [dynamic\\_bitset](#) ([dynamic\\_bitset](#) &&\_\_b) noexcept
- template<typename \_Traits = std::char\_traits<char>, typename \_CharT = typename \_Traits::char\_type>  
void **\_M\_copy\_from\_ptr** (const \_CharT \*, size\_t, size\_t, size\_t, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
void **\_M\_copy\_from\_string** (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, size\_t \_\_pos, size\_t \_\_n, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))
- template<typename \_CharT, typename \_Traits, typename \_Alloc1 >  
void **\_M\_copy\_to\_string** ([std::basic\\_string](#)< \_CharT, \_Traits, \_Alloc1 > &\_\_str, \_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1')) const
- bool [all](#) () const
- bool [any](#) () const
- void [append](#) (block\_type \_\_block)
- void **append** (initializer\_list< block\_type > \_\_il)
- template<typename \_BlockInputIterator >  
void [append](#) (\_BlockInputIterator \_\_first, \_BlockInputIterator \_\_last)
- void [clear](#) ()
- size\_type [count](#) () const noexcept
- bool [empty](#) () const noexcept
- size\_type [find\\_first](#) () const
- size\_type [find\\_next](#) (size\_t \_\_prev) const
- [dynamic\\_bitset](#) & [flip](#) ()
- [dynamic\\_bitset](#) & [flip](#) (size\_type \_\_pos)
- allocator\_type [get\\_allocator](#) () const noexcept
- bool **is\_proper\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const
- bool **is\_subset\_of** (const [dynamic\\_bitset](#) &\_\_b) const
- constexpr size\_type [max\\_size](#) () noexcept
- bool [none](#) () const
- size\_type [num\\_blocks](#) () const noexcept
- [dynamic\\_bitset](#) & [operator=](#) (const [dynamic\\_bitset](#) &)=default
- [dynamic\\_bitset](#) & [operator=](#) ([dynamic\\_bitset](#) &&\_\_b) noexcept([std::is\\_nothrow\\_move\\_assignable](#)< \_Base >::value)
- [dynamic\\_bitset](#) [operator~](#) () const
- void [push\\_back](#) (bool \_\_bit)
- [dynamic\\_bitset](#) & [reset](#) ()
- [dynamic\\_bitset](#) & [reset](#) (size\_type \_\_pos)
- void [resize](#) (size\_type \_\_nbits, bool \_\_value=false)
- [dynamic\\_bitset](#) & [set](#) ()

- `dynamic_bitset` & `set` (size\_type \_\_pos, bool \_\_val=true)
- size\_type `size` () const noexcept
- void `swap` (dynamic\_bitset &\_\_b) noexcept
- bool `test` (size\_type \_\_pos) const
- template<typename \_CharT = char, typename \_Traits = std::char\_traits<\_CharT>, typename \_Alloc1 = std::allocator<\_CharT>>  
std::basic\_string<\_CharT, \_Traits, \_Alloc1 > `to_string` (\_CharT \_\_zero=\_CharT('0'), \_CharT \_\_one=\_CharT('1'))  
const
- unsigned long long `to_ullong` () const
- unsigned long `to_ulong` () const

- `dynamic_bitset` & `operator&=` (const dynamic\_bitset &\_\_rhs)
- `dynamic_bitset` & `operator&=` (dynamic\_bitset &&\_\_rhs)
- `dynamic_bitset` & `operator|=` (const dynamic\_bitset &\_\_rhs)
- `dynamic_bitset` & `operator^=` (const dynamic\_bitset &\_\_rhs)
- `dynamic_bitset` & `operator-=` (const dynamic\_bitset &\_\_rhs)

- `dynamic_bitset` & `operator<<=` (size\_type \_\_pos)
- `dynamic_bitset` & `operator>>=` (size\_type \_\_pos)

- `reference operator[]` (size\_type \_\_pos)
- const\_reference `operator[]` (size\_type \_\_pos) const

- `dynamic_bitset operator<<` (size\_type \_\_pos) const
- `dynamic_bitset operator>>` (size\_type \_\_pos) const

#### Static Public Attributes

- static const size\_type **bits\_per\_block**
- static const size\_type **npos**

## Private Member Functions

- `size_t _M_are_all_aux ()` const noexcept
- `void _M_clear ()` noexcept
- `void _M_do_and (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_append_block (block_type __block, size_type __pos)`
- `size_t _M_do_count ()` const noexcept
- `void _M_do_dif (const __dynamic_bitset_base &__x)` noexcept
- `size_type _M_do_find_first (size_t __not_found)` const
- `size_type _M_do_find_next (size_t __prev, size_t __not_found)` const
- `void _M_do_flip ()` noexcept
- `void _M_do_left_shift (size_t __shift)`
- `void _M_do_or (const __dynamic_bitset_base &__x)` noexcept
- `void _M_do_reset ()` noexcept
- `void _M_do_right_shift (size_t __shift)`
- `void _M_do_set ()` noexcept
- `unsigned long long _M_do_to_ullong ()` const
- `unsigned long _M_do_to_ulong ()` const
- `void _M_do_xor (const __dynamic_bitset_base &__x)` noexcept
- `allocator_type _M_get_allocator ()` const noexcept
- `block_type & _M_getword (size_type __pos)` noexcept
- `block_type _M_getword (size_type __pos)` const noexcept
- `block_type & _M_hiword ()` noexcept
- `block_type _M_hiword ()` const noexcept
- `bool _M_is_any ()` const noexcept
- `bool _M_is_equal (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_less (const __dynamic_bitset_base &__x)` const noexcept
- `bool _M_is_proper_subset_of (const __dynamic_bitset_base &__b)` const noexcept
- `bool _M_is_subset_of (const __dynamic_bitset_base &__b)` noexcept
- `void _M_resize (size_t __nbits, bool __value)`
- `size_type _M_size ()` const noexcept
- `void _M_swap (__dynamic_bitset_base &__b)` noexcept

## Static Private Member Functions

- `static block_type _S_maskbit (size_type __pos)` noexcept
- `static size_type _S_whichbit (size_type __pos)` noexcept
- `static size_type _S_whichbyte (size_type __pos)` noexcept
- `static size_type _S_whichword (size_type __pos)` noexcept

## Private Attributes

- `std::vector< block_type, allocator_type > _M_w`

## Static Private Attributes

- `static const size_type _S_bits_per_block`

## Friends

- `bool operator<` (const `dynamic_bitset` &\_\_lhs, const `dynamic_bitset` &\_\_rhs) noexcept
- `bool operator==` (const `dynamic_bitset` &\_\_lhs, const `dynamic_bitset` &\_\_rhs) noexcept
- class `reference`

### 4.458.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >
```

The `dynamic_bitset` class represents a sequence of bits.

See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf>

In the general unoptimized case, storage is allocated in word-sized blocks. Let  $B$  be the number of bits in a word, then  $(Nb+(B-1))/B$  words will be used for storage.  $B - NbB$  bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.

If you think of `dynamic_bitset` as "a simple array of bits," be aware that your mental picture is reversed: a `dynamic_bitset` behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index  $Nb-1$  in the "most significant / left-hand" position. Thus, unlike other containers, a `dynamic_bitset`'s index "counts from right to left," to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>
using namespace std;
int main()
{
 long a = 'a';
 dynamic_bitset<> b(a);
 cout << "b('a') is " << b << endl;
 ostringstream s;
 s << b;
 string str = s.str();
 cout << "index 3 in the string is " << str[3] << " but\n"
 << "index 3 in the bitset is " << b[3] << endl;
}
```

Most of the actual code isn't contained in `dynamic_bitset<>` itself, but in the base class `__dynamic_bitset_base`. The base class works with whole words, not with individual bits. This allows us to specialize `__dynamic_bitset_base` for the important special case where the `dynamic_bitset` is only a single word.

Extra confusion can result due to the fact that the storage for `__dynamic_bitset_base` is a vector, and is indexed as such. This is carefully encapsulated.

Definition at line 419 of file `dynamic_bitset`.

### 4.458.2 Constructor & Destructor Documentation

## 4.458.2.1 dynamic\_bitset() [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset () [default]
```

All bits set to zero.

## 4.458.2.2 dynamic\_bitset() [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const allocator_type & __alloc) [inline], [explicit]
```

All bits set to zero.

Definition at line 578 of file dynamic\_bitset.

## 4.458.2.3 dynamic\_bitset() [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 size_type __nbits,
 unsigned long long __val = 0ULL,
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

Definition at line 584 of file dynamic\_bitset.

## 4.458.2.4 dynamic\_bitset() [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT , typename _Traits , typename _Alloc1 >
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
 typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
 typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<↵
 _CharT, _Traits, _Alloc1>::npos,
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1'),
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]
```

Use a subset of a string.

**Parameters**

|                      |                                                            |
|----------------------|------------------------------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters.                        |
| <code>__pos</code>   | Index of the first character in <code>__str</code> to use. |
| <code>__n</code>     | The number of characters to copy.                          |
| <code>__zero</code>  | The character to use for unset bits.                       |
| <code>__one</code>   | The character to use for set bits.                         |
| <code>__alloc</code> | An allocator.                                              |

**Exceptions**

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::out_of_range</code>     | If <code>__pos</code> is bigger the size of <code>__str</code> .   |
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |

Definition at line 609 of file `dynamic_bitset`.

**4.458.2.5 `dynamic_bitset()`** [5/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const char * __str,
 const allocator_type & __alloc = allocator_type()) [inline], [explicit]
```

Construct from a string.

**Parameters**

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__str</code>   | A string of '0' and '1' characters. |
| <code>__alloc</code> | An allocator.                       |

**Exceptions**

|                                    |                                                                    |
|------------------------------------|--------------------------------------------------------------------|
| <code>std::invalid_argument</code> | If a character appears in the string which is neither '0' nor '1'. |
|------------------------------------|--------------------------------------------------------------------|

Definition at line 636 of file `dynamic_bitset`.

**4.458.2.6 `dynamic_bitset()`** [6/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 const dynamic_bitset< _WordT, _Alloc > &) [default]
```

Copy constructor.



#### 4.458.2.7 dynamic\_bitset() [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
 dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move constructor.

Definition at line 648 of file dynamic\_bitset.

### 4.458.3 Member Function Documentation

#### 4.458.3.1 all()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all () const [inline]
```

Tests whether all the bits are on.

##### Returns

True if all the bits are set.

Definition at line 1036 of file dynamic\_bitset.

#### 4.458.3.2 any()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any () const [inline]
```

Tests whether any of the bits are on.

##### Returns

True if at least one bit is set.

Definition at line 1044 of file dynamic\_bitset.

#### 4.458.3.3 `append()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
 block_type __block) [inline]
```

Append a block.

Definition at line 726 of file `dynamic_bitset`.

#### 4.458.3.4 `append()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator >
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
 _BlockInputIterator __first,
 _BlockInputIterator __last) [inline]
```

Append an iterator range of blocks.

Definition at line 744 of file `dynamic_bitset`.

#### 4.458.3.5 `clear()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::clear () [inline]
```

Clear the bitset.

Definition at line 701 of file `dynamic_bitset`.

#### 4.458.3.6 `count()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::count () const [inline], [noexcept]
```

Returns the number of bits which are set.

Definition at line 992 of file `dynamic_bitset`.

#### 4.458.3.7 empty()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the dynamic\_bitset is empty.

Definition at line 1007 of file dynamic\_bitset.

#### 4.458.3.8 find\_first()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first () const [inline]
```

Finds the index of the first "on" bit.

##### Returns

The index of the first bit set, or size() if not found.

##### See also

find\_next

Definition at line 1072 of file dynamic\_bitset.

#### 4.458.3.9 find\_next()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (
 size_t __prev) const [inline]
```

Finds the index of the next "on" bit after prev.

##### Returns

The index of the next bit set, or size() if not found.

##### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__prev</code> | Where to start searching. |
|---------------------|---------------------------|

See also

[find\\_first](#)

Definition at line 1082 of file `dynamic_bitset`.

#### 4.458.3.10 `flip()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::flip () [inline]
```

Toggles every bit to its opposite value.

Definition at line 883 of file `dynamic_bitset`.

#### 4.458.3.11 `flip()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::flip (
 size_type __pos) [inline]
```

Toggles a given bit to its opposite value.

##### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

##### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Definition at line 896 of file `dynamic_bitset`.

#### 4.458.3.12 `get_allocator()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset< _WordT, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Return the allocator for the bitset.

Definition at line 681 of file `dynamic_bitset`.

## 4.458.3.13 max\_size()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
constexpr size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::max_size () [inline], [noexcept]
```

Returns the maximum size of a dynamic\_bitset object having the same type as \*this. The real answer is max() \* bits\_per\_block but is likely to overflow.

Definition at line 1014 of file dynamic\_bitset.

## 4.458.3.14 none()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset<_WordT, _Alloc >::none () const [inline]
```

Tests whether any of the bits are on.

## Returns

True if none of the bits are set.

Definition at line 1052 of file dynamic\_bitset.

## 4.458.3.15 num\_blocks()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset<_WordT, _Alloc >::num_blocks () const [inline], [noexcept]
```

Returns the total number of blocks.

Definition at line 1002 of file dynamic\_bitset.

## 4.458.3.16 operator&amp;=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator&= (
 const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

## Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

Definition at line 759 of file `dynamic_bitset`.

#### 4.458.3.17 `operator&=()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
 dynamic_bitset< _WordT, _Alloc > && __rhs) [inline]
```

Operations on `dynamic_bitsets`.

##### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

Definition at line 766 of file `dynamic_bitset`.

#### 4.458.3.18 `operator-=()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (
 const dynamic_bitset< _WordT, _Alloc > & __rhs) [inline]
```

Operations on `dynamic_bitsets`.

##### Parameters

|                    |                                            |
|--------------------|--------------------------------------------|
| <code>__rhs</code> | A same-sized <code>dynamic_bitset</code> . |
|--------------------|--------------------------------------------|

These should be self-explanatory.

Definition at line 787 of file `dynamic_bitset`.

#### 4.458.3.19 `operator<<()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (
 size_type __pos) const [inline]
```

Self-explanatory.

Definition at line 1058 of file `dynamic_bitset`.

## 4.458.3.20 operator&lt;&lt;=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
 size_type __pos) [inline]
```

Operations on dynamic\_bitsets.

## Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

Definition at line 802 of file dynamic\_bitset.

## 4.458.3.21 operator=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
 const dynamic_bitset< _WordT, _Alloc > &) [default]
```

Copy assignment operator.

## 4.458.3.22 operator=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
 dynamic_bitset< _WordT, _Alloc > && __b) [inline], [noexcept]
```

Move assignment operator.

Definition at line 665 of file dynamic\_bitset.

## 4.458.3.23 operator&gt;&gt;()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
 size_type __pos) const [inline]
```

Self-explanatory.

Definition at line 1062 of file dynamic\_bitset.

## 4.458.3.24 operator&gt;&gt;=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
 size_type __pos) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                                |
|--------------------|--------------------------------|
| <code>__pos</code> | The number of places to shift. |
|--------------------|--------------------------------|

These should be self-explanatory.

Definition at line 815 of file `dynamic_bitset`.

**4.458.3.25 operator[]()** [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
 size_type __pos) [inline]
```

Array-indexing support.

**Parameters**

|                    |                                              |
|--------------------|----------------------------------------------|
| <code>__pos</code> | Index into the <code>dynamic_bitset</code> . |
|--------------------|----------------------------------------------|

**Returns**

A bool for a 'const `dynamic_bitset`'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 918 of file `dynamic_bitset`.

**4.458.3.26 operator[]()** [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
 size_type __pos) const [inline]
```

Array-indexing support.

**Parameters**

|                    |                                              |
|--------------------|----------------------------------------------|
| <code>__pos</code> | Index into the <code>dynamic_bitset</code> . |
|--------------------|----------------------------------------------|



**Returns**

A bool for a 'const dynamic\_bitset'. For non-const bitsets, an instance of the reference proxy class.

**Note**

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

Definition at line 922 of file dynamic\_bitset.

**4.458.3.27 operator^=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator^= (
 const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

Definition at line 780 of file dynamic\_bitset.

**4.458.3.28 operator" |=()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::operator|= (
 const dynamic_bitset<_WordT, _Alloc > & __rhs) [inline]
```

Operations on dynamic\_bitsets.

**Parameters**

|                    |                              |
|--------------------|------------------------------|
| <code>__rhs</code> | A same-sized dynamic_bitset. |
|--------------------|------------------------------|

These should be self-explanatory.

Definition at line 773 of file dynamic\_bitset.

#### 4.458.3.29 `operator~()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset< std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ () const [inline]
```

See the no-argument `flip()`.

Definition at line 905 of file `dynamic_bitset`.

#### 4.458.3.30 `push_back()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
 bool __bit) [inline]
```

Push a bit onto the high end of the bitset.

Definition at line 711 of file `dynamic_bitset`.

#### 4.458.3.31 `reset()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset () [inline]
```

Sets every bit to false.

Definition at line 858 of file `dynamic_bitset`.

#### 4.458.3.32 `reset()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
 size_type __pos) [inline]
```

Sets a given bit to false.

##### Parameters

|                    |                       |
|--------------------|-----------------------|
| <code>__pos</code> | The index of the bit. |
|--------------------|-----------------------|

##### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Same as writing `set(__pos, false)`.

Definition at line 872 of file `dynamic_bitset`.

#### 4.458.3.33 `resize()`

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset<_WordT, _Alloc >::resize (
 size_type __nbits,
 bool __value = false) [inline]
```

Resize the bitset.

Definition at line 688 of file `dynamic_bitset`.

#### 4.458.3.34 `set()` [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::set () [inline]
```

Sets every bit to true.

Definition at line 833 of file `dynamic_bitset`.

#### 4.458.3.35 `set()` [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset& std::tr2::dynamic_bitset<_WordT, _Alloc >::set (
 size_type __pos,
 bool __val = true) [inline]
```

Sets a given bit to a particular value.

##### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__pos</code> | The index of the bit.                   |
| <code>__val</code> | Either true or false, defaults to true. |

##### Exceptions

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Definition at line 847 of file `dynamic_bitset`.

**4.458.3.36 size()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size () const [inline], [noexcept]
```

Returns the total number of bits.

Definition at line 997 of file `dynamic_bitset`.

Referenced by `std::tr2::operator>>()`.

**4.458.3.37 swap()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (
 dynamic_bitset< _WordT, _Alloc > & __b) [inline], [noexcept]
```

Swap with another bitset.

Definition at line 654 of file `dynamic_bitset`.

**4.458.3.38 test()**

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (
 size_type __pos) const [inline]
```

Tests the value of a bit.

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__pos</code> | The index of a bit. |
|--------------------|---------------------|

**Returns**

The value at `__pos`.

**Exceptions**

|                                |                                                      |
|--------------------------------|------------------------------------------------------|
| <code>std::out_of_range</code> | If <code>__pos</code> is bigger the size of the set. |
|--------------------------------|------------------------------------------------------|

Definition at line 1024 of file `dynamic_bitset`.

## 4.458.3.39 to\_string()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string<_CharT, _Traits, _Alloc1> std::tr2::dynamic_bitset<_WordT, _Alloc >::to_↵
string (
 _CharT __zero = _CharT('0'),
 _CharT __one = _CharT('1')) const [inline]
```

Returns a character interpretation of the dynamic\_bitset.

**Returns**

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

Definition at line 958 of file dynamic\_bitset.

## 4.458.3.40 to\_ulong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ulong () const [inline]
```

Returns a numerical interpretation of the dynamic\_bitset.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

Definition at line 943 of file dynamic\_bitset.

## 4.458.3.41 to\_ulong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset<_WordT, _Alloc >::to_ulong () const [inline]
```

Returns a numerical interpretation of the dynamic\_bitset.

**Returns**

The integral equivalent of the bits.

**Exceptions**

|                                  |                                                                   |
|----------------------------------|-------------------------------------------------------------------|
| <code>std::overflow_error</code> | If there are too many bits to be represented in an unsigned long. |
|----------------------------------|-------------------------------------------------------------------|

Definition at line 933 of file `dynamic_bitset`.

The documentation for this class was generated from the following files:

- [dynamic\\_bitset](#)
- [dynamic\\_bitset.tcc](#)

## 4.459 `std::enable_if< bool, _Tp >` Struct Template Reference

### 4.459.1 Detailed Description

```
template<bool, typename _Tp = void>
struct std::enable_if< bool, _Tp >
```

Define a member typedef `type` only if a boolean constant is true.

Definition at line 2182 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.460 `std::enable_shared_from_this< _Tp >` Class Template Reference

### Public Member Functions

- [shared\\_ptr< \\_Tp >](#) **shared\_from\_this** ()
- [shared\\_ptr< const \\_Tp >](#) **shared\_from\_this** () const
- [weak\\_ptr< \\_Tp >](#) **weak\_from\_this** () noexcept
- [weak\\_ptr< const \\_Tp >](#) **weak\_from\_this** () const noexcept

### Protected Member Functions

- **enable\_shared\_from\_this** (const [enable\\_shared\\_from\\_this](#) &) noexcept
- [enable\\_shared\\_from\\_this](#) & **operator=** (const [enable\\_shared\\_from\\_this](#) &) noexcept

## Friends

- `const enable\_shared\_from\_this * __enable_shared_from_this_base` (`const __shared_count<> &`, `const enable\_shared\_from\_this *__p`)
- `template<typename , _Lock_policy >`  
`class __shared_ptr`

## 4.460.1 Detailed Description

```
template<typename _Tp>
class std::enable_shared_from_this<_Tp>
```

Base class allowing use of member function `shared_from_this`.

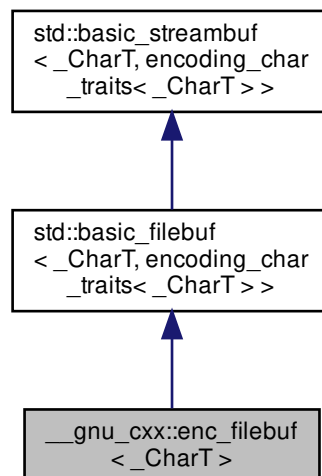
Definition at line 791 of file `bits/shared_ptr.h`.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

4.461 `__gnu_cxx::enc_filebuf<_CharT>` Class Template Reference

Inheritance diagram for `__gnu_cxx::enc_filebuf<_CharT>`:



## Public Types

- typedef codecvt< [char\\_type](#), char, \_\_state\_type > **\_\_codecvt\_type**
- typedef \_\_basic\_file< char > **\_\_file\_type**
- typedef [basic\\_filebuf](#)< [char\\_type](#), [traits\\_type](#) > **\_\_filebuf\_type**
- typedef traits\_type::state\_type **\_\_state\_type**
- typedef [basic\\_streambuf](#)< [char\\_type](#), [traits\\_type](#) > **\_\_streambuf\_type**
- typedef \_CharT **char\_type**
- typedef traits\_type::int\_type **int\_type**
- typedef traits\_type::off\_type **off\_type**
- typedef traits\_type::pos\_type **pos\_type**
- typedef traits\_type::state\_type **state\_type**
- typedef [encoding\\_char\\_traits](#)< \_CharT > **traits\_type**

## Public Member Functions

- **enc\_filebuf** (state\_type &\_\_state)
  - [\\_\\_filebuf\\_type](#) \* **close** ()
  - locale [getloc](#) () const
  - streamsize [in\\_avail](#) ()
  - bool [is\\_open](#) () const throw ()
  - [\\_\\_filebuf\\_type](#) \* **open** (const char \*\_\_s, ios\_base::openmode \_\_mode)
  - [\\_\\_filebuf\\_type](#) \* **open** (const [std::string](#) &\_\_s, ios\_base::openmode \_\_mode)
  - locale [pubimbue](#) (const locale &\_\_loc)
  - [int\\_type](#) **sbumpc** ()
  - [int\\_type](#) **sgetc** ()
  - streamsize [sgetn](#) ([char\\_type](#) \*\_\_s, streamsize \_\_n)
  - [int\\_type](#) **snextc** ()
  - [int\\_type](#) **sputbackc** ([char\\_type](#) \_\_c)
  - [int\\_type](#) **sputc** ([char\\_type](#) \_\_c)
  - streamsize **sputn** (const [char\\_type](#) \*\_\_s, streamsize \_\_n)
  - [int\\_type](#) **sungetc** ()
  - void **swap** ([basic\\_filebuf](#) &)
- 
- [basic\\_streambuf](#) \* **pubsetbuf** ([char\\_type](#) \*\_\_s, streamsize \_\_n)
  - [pos\\_type](#) **pubseekoff** ([off\\_type](#) \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - [pos\\_type](#) **pubseekpos** ([pos\\_type](#) \_\_sp, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - int **pubsync** ()



## Protected Member Functions

- `void __safe_gbump (streamsize __n)`
  - `void __safe_pbump (streamsize __n)`
  - `void _M_allocate_internal_buffer ()`
  - `bool _M_convert_to_external (char_type *, streamsize)`
  - `void _M_create_pback ()`
  - `void _M_destroy_internal_buffer () throw ()`
  - `void _M_destroy_pback () throw ()`
  - `int _M_get_ext_pos (__state_type &__state)`
  - `pos_type _M_seek (off_type __off, ios_base::seekdir __way, __state_type __state)`
  - `void _M_set_buffer (streamsize __off)`
  - `bool _M_terminate_output ()`
  - `void gbump (int __n)`
  - `virtual void imbue (const locale &__loc)`
  - `virtual int_type overflow (int_type __c=encoding_char_traits<_CharT>::eof())`
  - `virtual int_type pbackfail (int_type __c=encoding_char_traits<_CharT>::eof())`
  - `void pbump (int __n)`
  - `virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)`
  - `void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)`
  - `void setp (char_type *__pbeg, char_type *__pend)`
  - `virtual streamsize showmanyc ()`
  - `void swap (basic_streambuf &__sb)`
  - `virtual int sync ()`
  - `virtual int_type uflow ()`
  - `virtual int_type underflow ()`
  - `virtual streamsize xsgetn (char_type *__s, streamsize __n)`
  - `virtual streamsize xsputn (const char_type *__s, streamsize __n)`
- 
- `char_type * eback () const`
  - `char_type * gptr () const`
  - `char_type * egptr () const`
- 
- `char_type * pbase () const`
  - `char_type * pptr () const`
  - `char_type * epptr () const`

### Protected Attributes

- [char\\_type](#) \* [\\_M\\_buf](#)
  - bool [\\_M\\_buf\\_allocated](#)
  - locale [\\_M\\_buf\\_locale](#)
  - size\_t [\\_M\\_buf\\_size](#)
  - const [\\_\\_codecvt\\_type](#) \* [\\_M\\_codecvt](#)
  - char \* [\\_M\\_ext\\_buf](#)
  - streamsize [\\_M\\_ext\\_buf\\_size](#)
  - char \* [\\_M\\_ext\\_end](#)
  - const char \* [\\_M\\_ext\\_next](#)
  - [\\_\\_file\\_type](#) [\\_M\\_file](#)
  - [char\\_type](#) \* [\\_M\\_in\\_beg](#)
  - [char\\_type](#) \* [\\_M\\_in\\_cur](#)
  - [char\\_type](#) \* [\\_M\\_in\\_end](#)
  - [\\_\\_c\\_lock](#) [\\_M\\_lock](#)
  - ios\_base::openmode [\\_M\\_mode](#)
  - [char\\_type](#) \* [\\_M\\_out\\_beg](#)
  - [char\\_type](#) \* [\\_M\\_out\\_cur](#)
  - [char\\_type](#) \* [\\_M\\_out\\_end](#)
  - bool [\\_M\\_reading](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_beg](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_cur](#)
  - [\\_\\_state\\_type](#) [\\_M\\_state\\_last](#)
  - bool [\\_M\\_writing](#)
- 
- [char\\_type](#) [\\_M\\_pback](#)
  - [char\\_type](#) \* [\\_M\\_pback\\_cur\\_save](#)
  - [char\\_type](#) \* [\\_M\\_pback\\_end\\_save](#)
  - bool [\\_M\\_pback\\_init](#)

#### 4.461.1 Detailed Description

```
template<typename _CharT>
class __gnu_cxx::enc_filebuf< _CharT >
```

class enc\_filebuf.

Definition at line 42 of file enc\_filebuf.h.

#### 4.461.2 Member Function Documentation

4.461.2.1 `_M_create_pback()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_create_pback () [inline],
[protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

4.461.2.2 `_M_destroy_pback()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_destroy_pback () throw (
) [inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

4.461.2.3 `_M_set_buffer()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_set_buffer (
 streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 459 of file `fstream`.

4.461.2.4 `close()`

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__filebuf_type * std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::close () [inherited]
```

Closes the currently associated file.

## Returns

`this` on success, `NULL` on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 250 of file `fstream.tcc`.

#### 4.461.2.5 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 489 of file streambuf.

#### 4.461.2.6 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 4.461.2.7 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 4.461.2.8 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Definition at line 505 of file `streambuf`.

4.461.2.9 `getloc()`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]
```

Locale access.

## Returns

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file `streambuf`.

4.461.2.10 `gptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file `streambuf`.

4.461.2.11 `imbue()`

```
void std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::imbue (
 const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from `streambuf` can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 1030 of file `fstream.tcc`.

**4.461.2.12 in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file `streambuf`.

**4.461.2.13 is\_open()**

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::is_open () const throw ()
[inline], [inherited]
```

Returns true if the external file is open.

Definition at line 265 of file `fstream`.

**4.461.2.14 open() [1/2]**

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↵
_Chart, encoding_char_traits< _CharT > >::open (
 const char * __s,
 ios_base::openmode __mode) [inherited]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           |    | +   |       | +   | a+               |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

Definition at line 180 of file `fstream.tcc`.

4.461.2.15 `open()` [2/2]

```
__filebuf_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::open (
 const std::string & __s,
 ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

**Parameters**

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

**Returns**

`this` on success, `NULL` on failure

Definition at line 331 of file `fstream`.

**4.461.2.16 overflow()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::overflow (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 542 of file `fstream.tcc`.

**4.461.2.17 pbackfail()**

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::int_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.



## Parameters

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_c</code> | The character to be inserted back into the sequence. |
|-----------------|------------------------------------------------------|

## Returns

`eof()` on failure, *some other value* on success

## Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 483 of file `fstream.tcc`.

4.461.2.18 `pbase()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

4.461.2.19 `pbump()`

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

**4.461.2.20 pptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

**4.461.2.21 pubimbue()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

**Parameters**

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

**Returns**

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 4.461.2.22 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 4.461.2.23 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

## 4.461.2.24 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 4.461.2.25 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync()`.

#### 4.461.2.26 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc () [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.461.2.27 `seekoff()`

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 833 of file `fstream.tcc`.

## 4.461.2.28 seekpos()

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::pos_type std::basic_filebuf<_CharT,
encoding_char_traits<_CharT>>::seekpos (
 pos_type __pos,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

## Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 893 of file `fstream.tcc`.

## 4.461.2.29 setbuf()

```
basic_filebuf<_CharT, encoding_char_traits<_CharT>>::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits<_CharT>>::setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

## Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__s</code> | Pointer to a buffer area.  |
| <code>__n</code> | Size of <code>__s</code> . |

## Returns

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 804 of file `fstream.tcc`.

**4.461.2.30 setg()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

**Parameters**

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

**Postcondition**

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

**4.461.2.31 setp()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

**Parameters**

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

**Postcondition**

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

## 4.461.2.32 sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

## 4.461.2.33 sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

## Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

## 4.461.2.34 showmanyc()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc () [protected],
[virtual], [inherited]
```

Investigating the data available.

### Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

### Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 298 of file `fstream.tcc`.

#### 4.461.2.35 `snextc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

### Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

#### 4.461.2.36 `sputbackc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|



**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.461.2.37 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

**4.461.2.38 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**4.461.2.39 `sungetc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**4.461.2.40 `sync()`**

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync (
 void) [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 1013 of file `fstream.tcc`.

## 4.461.2.41 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>`.

Definition at line 707 of file `streambuf`.

## 4.461.2.42 underflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::underflow () [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 324 of file `fstream.tcc`.

## 4.461.2.43 xsgetn()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 670 of file `fstream.tcc`.

**4.461.2.44 xsputn()**

```
streamsize std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 756 of file `fstream.tcc`.

### 4.461.3 Member Data Documentation

#### 4.461.3.1 \_M\_buf

```
char_type* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf [protected],
[inherited]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file fstream.

#### 4.461.3.2 \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file streambuf.

#### 4.461.3.3 \_M\_buf\_size

```
size_t std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_buf_size [protected],
[inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file fstream.

#### 4.461.3.4 \_M\_ext\_buf

```
char* std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_ext_buf [protected],
[inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file fstream.

#### 4.461.3.5 `_M_ext_buf_size`

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected],
[inherited]
```

Size of buffer held by `_M_ext_buf`.

Definition at line 183 of file `fstream`.

#### 4.461.3.6 `_M_ext_next`

```
const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected],
[inherited]
```

Pointers into the buffer held by `_M_ext_buf` that delimit a subsequence of bytes that have been read but not yet converted. When valid, `_M_ext_next` corresponds to `egptr()`.

Definition at line 190 of file `fstream`.

#### 4.461.3.7 `_M_in_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file `streambuf`.

#### 4.461.3.8 `_M_in_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file `streambuf`.

#### 4.461.3.9 `_M_in_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file `streambuf`.

#### 4.461.3.10 \_M\_mode

```
ios_base::openmode std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_mode [protected],
[inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file fstream.

#### 4.461.3.11 \_M\_out\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.461.3.12 \_M\_out\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.461.3.13 \_M\_out\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

#### 4.461.3.14 \_M\_pback

```
char_type std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_pback [protected],
[inherited]
```

Necessary bits for putback buffer management.

#### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file fstream.

#### 4.461.3.15 `_M_pback_cur_save`

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_cur_save [protected],
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 4.461.3.16 `_M_pback_end_save`

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected],
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

#### 4.461.3.17 `_M_pback_init`

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init [protected],
[inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 167 of file fstream.



## 4.461.3.18 \_M\_reading

```
bool std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>::_M_reading [protected],
[inherited]
```

\_M\_reading == false && \_M\_writing == false for **uncommitted** mode; \_M\_reading == true for **read** mode; \_M\_writing == true for **write** mode;

NB: \_M\_reading == true && \_M\_writing == true is unused.

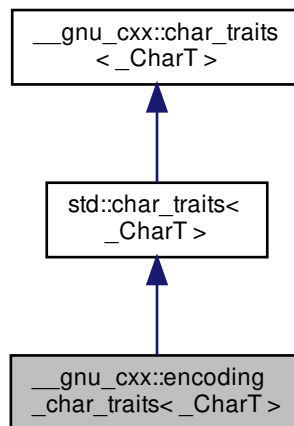
Definition at line 155 of file fstream.

The documentation for this class was generated from the following file:

- [enc\\_filebuf.h](#)

## 4.462 \_\_gnu\_cxx::encoding\_char\_traits&lt;\_CharT&gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_cxx::encoding\_char\_traits<\_CharT>:



## Public Types

- typedef \_CharT **char\_type**
- typedef [\\_Char\\_types](#)<\_CharT>::int\_type **int\_type**
- typedef [\\_Char\\_types](#)<\_CharT>::off\_type **off\_type**
- typedef [std::fpos](#)<state\_type> **pos\_type**
- typedef [encoding\\_state](#) **state\_type**

### Static Public Member Functions

- static constexpr void **assign** (char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **assign** (char\_type \* \_\_s, std::size\_t \_\_n, char\_type \_\_a)
- static constexpr int **compare** (const char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr char\_type \* **copy** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **eof** ()
- static constexpr bool **eq** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr bool **eq\_int\_type** (const int\_type &\_\_c1, const int\_type &\_\_c2)
- static constexpr const char\_type \* **find** (const char\_type \* \_\_s, std::size\_t \_\_n, const char\_type &\_\_a)
- static constexpr std::size\_t **length** (const char\_type \* \_\_s)
- static constexpr bool **lt** (const char\_type &\_\_c1, const char\_type &\_\_c2)
- static constexpr char\_type \* **move** (char\_type \* \_\_s1, const char\_type \* \_\_s2, std::size\_t \_\_n)
- static constexpr int\_type **not\_eof** (const int\_type &\_\_c)
- static constexpr char\_type **to\_char\_type** (const int\_type &\_\_c)
- static constexpr int\_type **to\_int\_type** (const char\_type &\_\_c)

#### 4.462.1 Detailed Description

```
template<typename _CharT>
struct __gnu_cxx::encoding_char_traits< _CharT >
```

encoding\_char\_traits

Definition at line 211 of file codecvt\_specializations.h.

The documentation for this struct was generated from the following file:

- [codecvt\\_specializations.h](#)

#### 4.463 \_\_gnu\_cxx::encoding\_state Class Reference

##### Public Types

- typedef iconv\_t **descriptor\_type**

##### Public Member Functions

- **encoding\_state** (const char \* \_\_int, const char \* \_\_ext, int \_\_ibom=0, int \_\_ebom=0, int \_\_bytes=1)
- **encoding\_state** (const [encoding\\_state](#) & \_\_obj)
- int **character\_ratio** () const
- int **external\_bom** () const
- const [std::string](#) **external\_encoding** () const
- bool **good** () const throw ()
- const descriptor\_type & **in\_descriptor** () const
- int **internal\_bom** () const
- const [std::string](#) **internal\_encoding** () const
- [encoding\\_state](#) & **operator=** (const [encoding\\_state](#) & \_\_obj)
- const descriptor\_type & **out\_descriptor** () const

#### Protected Member Functions

- void **construct** (const [encoding\\_state](#) &\_\_obj)
- void **destroy** () throw ()
- void **init** ()

#### Protected Attributes

- int **\_M\_bytes**
- int **\_M\_ext\_bom**
- [std::string](#) **\_M\_ext\_enc**
- descriptor\_type **\_M\_in\_desc**
- int **\_M\_int\_bom**
- [std::string](#) **\_M\_int\_enc**
- descriptor\_type **\_M\_out\_desc**

#### 4.463.1 Detailed Description

Extension to use iconv for dealing with character encodings.

Definition at line 51 of file `codecv_t_specializations.h`.

The documentation for this class was generated from the following file:

- [codecv\\_t\\_specializations.h](#)

## 4.464 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>` Struct Template Reference

#### 4.464.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, No_Throw>
```

Entry compare, primary template.

Definition at line 50 of file `entry_cmp.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

## 4.465 `__gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>` Struct Template Reference

#### Classes

- struct [type](#)

## Public Types

- typedef \_\_rebind\_v::const\_pointer **entry**

### 4.465.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >
```

Specialization, false.

Definition at line 62 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

### 4.466 \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true > Struct Template Reference

## Public Types

- typedef Cmp\_Fn [type](#)

### 4.466.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry\_cmp.hpp.

### 4.466.2 Member Typedef Documentation

#### 4.466.2.1 type

```
template<typename _VTp , typename Cmp_Fn , typename _Alloc >
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
```

Compare.

Definition at line 57 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)

4.467 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw >` Struct Template Reference

## 4.467.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, No_Throw >
```

Entry predicate primary class template.

Definition at line 50 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

4.468 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false >` Struct Template Reference

## Public Types

- typedef `__rebind_v::const_pointer` **entry**

## 4.468.1 Detailed Description

```
template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, false >
```

Specialization, `false`.

Definition at line 61 of file `entry_pred.hpp`.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

4.469 `__gnu_pbds::detail::entry_pred<_VTp, Pred, _Alloc, true >` Struct Template Reference

## Public Types

- typedef `Pred` **type**

#### 4.469.1 Detailed Description

```
template<typename VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< VTp, Pred, _Alloc, true >
```

Specialization, true.

Definition at line 54 of file entry\_pred.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_pred.hpp](#)

#### 4.470 \_\_gnu\_pbds::detail::eq\_by\_less< Key, Cmp\_Fn > Struct Template Reference

Inherits Cmp\_Fn.

##### Public Member Functions

- bool **operator()** (const Key &r\_lhs, const Key &r\_rhs) const

#### 4.470.1 Detailed Description

```
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >
```

Equivalence function.

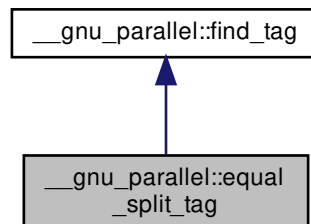
Definition at line 56 of file eq\_by\_less.hpp.

The documentation for this struct was generated from the following file:

- [eq\\_by\\_less.hpp](#)

#### 4.471 \_\_gnu\_parallel::equal\_split\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::equal\_split\_tag:



## 4.471.1 Detailed Description

Selects the equal splitting variant for std::find().

See also

`_GLIBCXX_FIND_EQUAL_SPLIT`

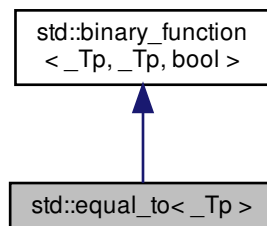
Definition at line 182 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.472 std::equal\_to&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::equal\_to< \_Tp >:



## Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

## Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 4.472.1 Detailed Description

```
template<typename _Tp>
struct std::equal_to<_Tp>
```

One of the [comparison functors](#).

Definition at line 331 of file `stl_function.h`.

#### 4.472.2 Member Typedef Documentation

##### 4.472.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.472.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.472.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



4.473 `std::equal_to< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp`, typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`←  
`_t`)==`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)==`std::forward`< `_Up` >(`_u`))

## 4.473.1 Detailed Description

```
template<>
struct std::equal_to< void >
```

One of the [comparison functors](#).

Definition at line 488 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.474 `std::_V2::error_category` Class Reference

## Public Member Functions

- **error\_category** (const [error\\_category](#) &)=delete
- virtual [error\\_condition](#) **default\_error\_condition** (int `__i`) const noexcept
- virtual bool **equivalent** (int `__i`, const [error\\_condition](#) &`__cond`) const noexcept
- virtual bool **equivalent** (const [error\\_code](#) &`__code`, int `__i`) const noexcept
- virtual [string](#) **message** (int) const =0
- virtual const char \* **name** () const noexcept=0
- bool **operator!=** (const [error\\_category](#) &`__other`) const noexcept
- bool **operator<** (const [error\\_category](#) &`__other`) const noexcept
- [error\\_category](#) & **operator=** (const [error\\_category](#) &)=delete
- bool **operator==** (const [error\\_category](#) &`__other`) const noexcept

#### 4.474.1 Detailed Description

Abstract base class for types defining a category of error codes.

An error category defines a context that give meaning to the integer stored in an `error_code` or `error_condition` object. For example, the standard `errno` constants such a `EINVAL` and `ENOMEM` are associated with the "generic" category and other OS-specific error numbers are associated with the "system" category, but a user-defined category might give different meanings to the same numerical values.

Definition at line 89 of file `system_error`.

The documentation for this class was generated from the following file:

- [system\\_error](#)

#### 4.475 std::error\_code Struct Reference

##### Public Member Functions

- **error\_code** (int \_\_v, const error\_category &\_\_cat) noexcept
- template<typename \_ErrorCodeEnum , typename = typename enable\_if<is\_error\_code\_enum<\_ErrorCodeEnum>::value>::type> **error\_code** (\_ErrorCodeEnum \_\_e) noexcept
- void **assign** (int \_\_v, const error\_category &\_\_cat) noexcept
- const error\_category & **category** () const noexcept
- void **clear** () noexcept
- [error\\_condition](#) **default\_error\_condition** () const noexcept
- \_GLIBCXX\_DEFAULT\_ABI\_TAG [string](#) **message** () const
- **operator bool** () const noexcept
- template<typename \_ErrorCodeEnum > [enable\\_if](#)< [is\\_error\\_code\\_enum](#)< \_ErrorCodeEnum >::value, [error\\_code](#) & >::type **operator=** (\_ErrorCodeEnum \_\_e) noexcept
- int **value** () const noexcept

##### Related Functions

(Note that these are not member functions.)

- [error\\_condition](#) **make\_error\_condition** (errc) noexcept
- bool **operator!=** (const [error\\_code](#) &\_\_lhs, const [error\\_code](#) &\_\_rhs) noexcept
- bool **operator==** (const [error\\_code](#) &\_\_lhs, const [error\\_code](#) &\_\_rhs) noexcept
- [error\\_code](#) **make\_error\_code** (errc \_\_e) noexcept

## 4.475.1 Detailed Description

Class `error_code`

This class is a value type storing an integer error number and a category that gives meaning to the error number. Typically this is done close to the point where the error happens, to capture the original error value.

An `error_code` object can be used to store the original error value emitted by some subsystem, with a category relevant to the subsystem. For example, errors from POSIX library functions can be represented by an `errno` value and the "generic" category, but errors from an HTTP library might be represented by an HTTP response status code (e.g. 404) and a custom category defined by the library.

Definition at line 180 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

4.476 `std::error_condition` Struct Reference

## Public Member Functions

- **`error_condition`** (int \_\_v, const `error_category` &\_\_cat) noexcept
- template<typename `_ErrorConditionEnum` , typename = typename `enable_if`<`is_error_condition_enum`<`_ErrorConditionEnum`>::value>::type>  
**`error_condition`** (`_ErrorConditionEnum` \_\_e) noexcept
- void **`assign`** (int \_\_v, const `error_category` &\_\_cat) noexcept
- const `error_category` & **`category`** () const noexcept
- void **`clear`** () noexcept
- `_GLIBCXX_DEFAULT_ABI_TAG` **`string message`** () const
- **`operator bool`** () const noexcept
- template<typename `_ErrorConditionEnum` >  
`enable_if`<`is_error_condition_enum`<`_ErrorConditionEnum`>::value, `error_condition` & >::type **`operator=`** (`_ErrorConditionEnum` \_\_e) noexcept
- int **`value`** () const noexcept

## Related Functions

(Note that these are not member functions.)

- **`error_condition make_error_condition`** (errc \_\_e) noexcept
- bool **`operator!=`** (const `error_code` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- bool **`operator!=`** (const `error_condition` &\_\_lhs, const `error_code` &\_\_rhs) noexcept
- bool **`operator!=`** (const `error_condition` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- bool **`operator<`** (const `error_condition` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- bool **`operator==`** (const `error_code` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- bool **`operator==`** (const `error_condition` &\_\_lhs, const `error_condition` &\_\_rhs) noexcept
- bool **`operator==`** (const `error_condition` &\_\_lhs, const `error_code` &\_\_rhs) noexcept

#### 4.476.1 Detailed Description

Class `error_condition`

This class represents error conditions that may be visible at an API boundary. Different `error_code` values that can occur within a library or module might map to the same `error_condition`.

An `error_condition` represents something that the program can test for, and subsequently take appropriate action.

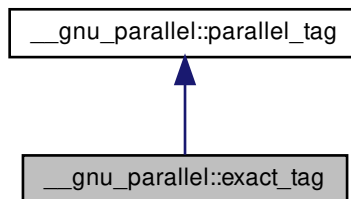
Definition at line 278 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 4.477 \_\_gnu\_parallel::exact\_tag Struct Reference

Inheritance diagram for `__gnu_parallel::exact_tag`:



#### Public Member Functions

- **exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) [\\_\\_get\\_num\\_threads](#) ()
- void [set\\_num\\_threads](#) ([\\_ThreadIndex](#) \_\_num\_threads)

#### 4.477.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 109 of file `tags.h`.

#### 4.477.2 Member Function Documentation

##### 4.477.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 4.477.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                                   |                            |
|-----------------------------------|----------------------------|
| <code><i>__num_threads</i></code> | Desired number of threads. |
|-----------------------------------|----------------------------|

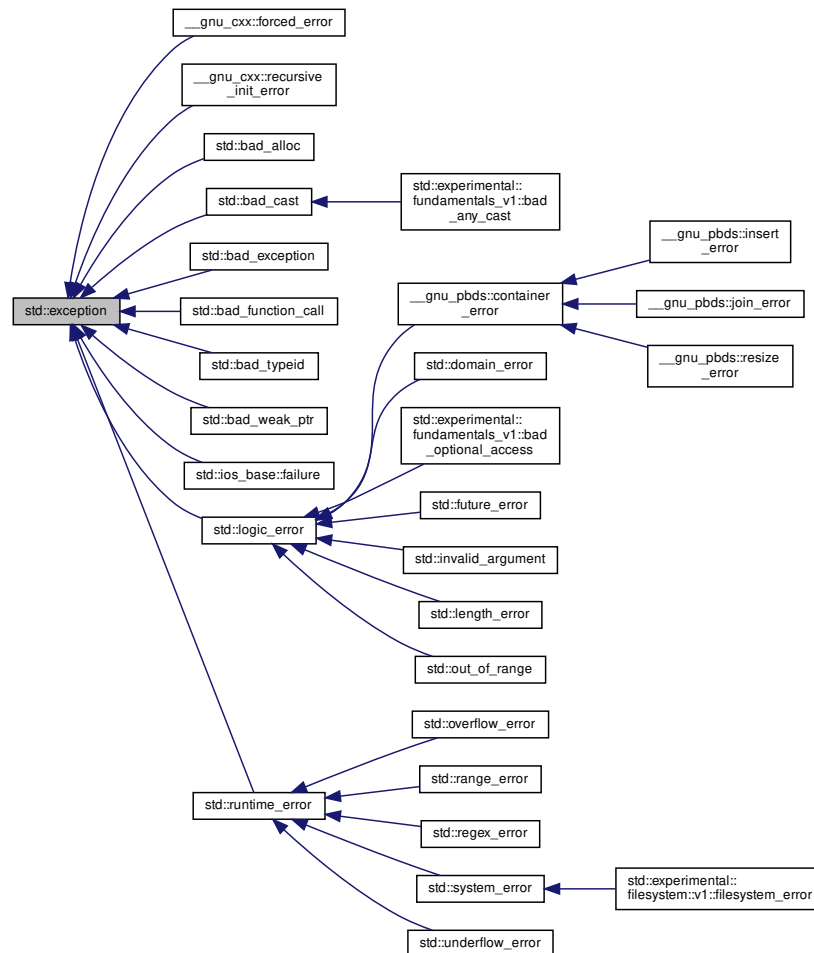
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.478 std::exception Class Reference

Inheritance diagram for std::exception:



### Public Member Functions

- **exception** (const [exception](#) &)=default
- **exception** ([exception](#) &&)=default
- [exception](#) & **operator=** (const [exception](#) &)=default
- [exception](#) & **operator=** ([exception](#) &&)=default
- virtual const char \* [what](#) () const noexcept

#### 4.478.1 Detailed Description

Base class for all library exceptions.

This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

Definition at line 60 of file `exception.h`.

The documentation for this class was generated from the following file:

- [exception.h](#)

## 4.479 `std::__exception_ptr::exception_ptr` Class Reference

### Public Member Functions

- **exception\_ptr** (const [exception\\_ptr](#) &) noexcept
- **exception\_ptr** (nullptr\_t) noexcept
- **exception\_ptr** ([exception\\_ptr](#) &&\_\_o) noexcept
- const class [std::type\\_info](#) \* **\_\_cxa\_exception\_type** () const noexcept
- **operator bool** () const
- [exception\\_ptr](#) & **operator=** (const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) & **operator=** ([exception\\_ptr](#) &&\_\_o) noexcept
- void **swap** ([exception\\_ptr](#) &) noexcept

### Friends

- bool **operator==** (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept
- [exception\\_ptr](#) **std::current\_exception** () noexcept
- template<typename \_Ex >  
[exception\\_ptr](#) **std::make\_exception\_ptr** (\_Ex) noexcept
- void **std::rethrow\_exception** ([exception\\_ptr](#))

### Related Functions

(Note that these are not member functions.)

- bool **operator==** (const [exception\\_ptr](#) &, const [exception\\_ptr](#) &) noexcept

#### 4.479.1 Detailed Description

An opaque pointer to an arbitrary exception.

Definition at line 80 of file `exception_ptr.h`.

#### 4.479.2 Friends And Related Function Documentation

##### 4.479.2.1 operator==()

```
bool operator== (
 const exception_ptr & ,
 const exception_ptr &) [related]
```

The documentation for this class was generated from the following file:

- [exception\\_ptr.h](#)

#### 4.480 std::exponential\_distribution<\_RealType> Class Template Reference

##### Classes

- struct [param\\_type](#)

##### Public Types

- typedef \_RealType [result\\_type](#)

##### Public Member Functions

- [exponential\\_distribution](#) ()
- [exponential\\_distribution](#) (\_RealType \_\_lambda)
- **exponential\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType [lambda](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) [operator\(\)](#) (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()



## Friends

- bool `operator==` (const `exponential_distribution` &\_\_d1, const `exponential_distribution` &\_\_d2)

## 4.480.1 Detailed Description

```
template<typename _RealType = double>
class std::exponential_distribution<_RealType>
```

An exponential continuous distribution for random numbers.

The formula for the exponential probability density function is  $p(x|\lambda) = \lambda e^{-\lambda x}$ .

**Table 1934 Distribution Statistics**

|                    |                         |
|--------------------|-------------------------|
| Mean               | $\frac{1}{\lambda}$     |
| Median             | $\frac{\ln 2}{\lambda}$ |
| Mode               | <i>zero</i>             |
| Range              | $[0, \infty]$           |
| Standard Deviation | $\frac{1}{\lambda}$     |

Definition at line 4645 of file random.h.

## 4.480.2 Member Typedef Documentation

## 4.480.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::exponential_distribution<_RealType>::result_type
```

The type of the range of the distribution.

Definition at line 4648 of file random.h.

## 4.480.3 Constructor &amp; Destructor Documentation

#### 4.480.3.1 `exponential_distribution()` [1/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution () [inline]
```

Constructs an exponential distribution with inverse scale parameter 1.0.

Definition at line 4689 of file random.h.

#### 4.480.3.2 `exponential_distribution()` [2/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution (
 _RealType __lambda) [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter  $\lambda$ .

Definition at line 4696 of file random.h.

### 4.480.4 Member Function Documentation

#### 4.480.4.1 `lambda()`

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda () const [inline]
```

Returns the inverse scale parameter of the distribution.

Definition at line 4717 of file random.h.

#### 4.480.4.2 `max()`

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4746 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

#### 4.480.4.3 min()

```
template<typename _RealType = double>
result_type std::exponential_distribution<_RealType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4739 of file random.h.

#### 4.480.4.4 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::exponential_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 4754 of file random.h.

#### 4.480.4.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4724 of file random.h.

Referenced by std::operator>>().

#### 4.480.4.6 param() [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution<_RealType>::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4732 of file random.h.

#### 4.480.4.7 reset()

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Has no effect on exponential distributions.

Definition at line 4711 of file random.h.

### 4.480.5 Friends And Related Function Documentation

#### 4.480.5.1 operator==

```
template<typename _RealType = double>
bool operator== (
 const exponential_distribution< _RealType > & __d1,
 const exponential_distribution< _RealType > & __d2) [friend]
```

Return true if two exponential distributions have the same parameters.

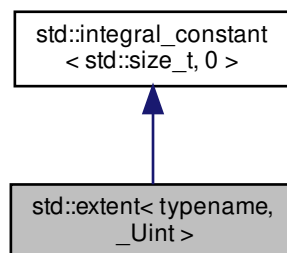
Definition at line 4794 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.481 std::extent< typename, \_UInt > Struct Template Reference

Inheritance diagram for std::extent< typename, \_UInt >:



### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr std::size\_t **value**

#### 4.481.1 Detailed Description

```
template<typename, unsigned _Uint>
struct std::extent< typename, _Uint >
```

extent

Definition at line 782 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.482 `std::extreme_value_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **extreme\_value\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **extreme\_value\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [extreme\\_value\\_distribution](#) &\_\_d1, const [extreme\\_value\\_distribution](#) &\_\_d2)

## 4.482.1 Detailed Description

```
template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >
```

A [extreme\\_value\\_distribution](#) random number distribution.

The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b} \exp\left(\frac{a-x}{b}\right) - \exp\left(\frac{a-x}{b}\right)$$

Definition at line 5070 of file random.h.

## 4.482.2 Member Typedef Documentation

#### 4.482.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::extreme_value_distribution<_RealType>::result_type
```

The type of the range of the distribution.

Definition at line 5073 of file random.h.

#### 4.482.3 Member Function Documentation

##### 4.482.3.1 a()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution<_RealType>::a () const [inline]
```

Return the  $a$  parameter of the distribution.

Definition at line 5135 of file random.h.

##### 4.482.3.2 b()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution<_RealType>::b () const [inline]
```

Return the  $b$  parameter of the distribution.

Definition at line 5142 of file random.h.

##### 4.482.3.3 max()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution<_RealType>::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5171 of file random.h.

References std::numeric\_limits<\_Tp>::max().

#### 4.482.3.4 min()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5164 of file random.h.

References `std::numeric_limits< _Tp >::lowest()`.

#### 4.482.3.5 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::extreme_value_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 5179 of file random.h.

#### 4.482.3.6 param() [1/2]

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5149 of file random.h.

Referenced by `std::operator>>()`.

#### 4.482.3.7 param() [2/2]

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|



Definition at line 5157 of file random.h.

#### 4.482.3.8 `reset()`

```
template<typename _RealType = double>
void std::extreme_value_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

Definition at line 5128 of file random.h.

### 4.482.4 Friends And Related Function Documentation

#### 4.482.4.1 `operator==`

```
template<typename _RealType = double>
bool operator== (
 const extreme_value_distribution<_RealType> & __d1,
 const extreme_value_distribution<_RealType> & __d2) [friend]
```

Return true if two extreme value distributions have the same parameters.

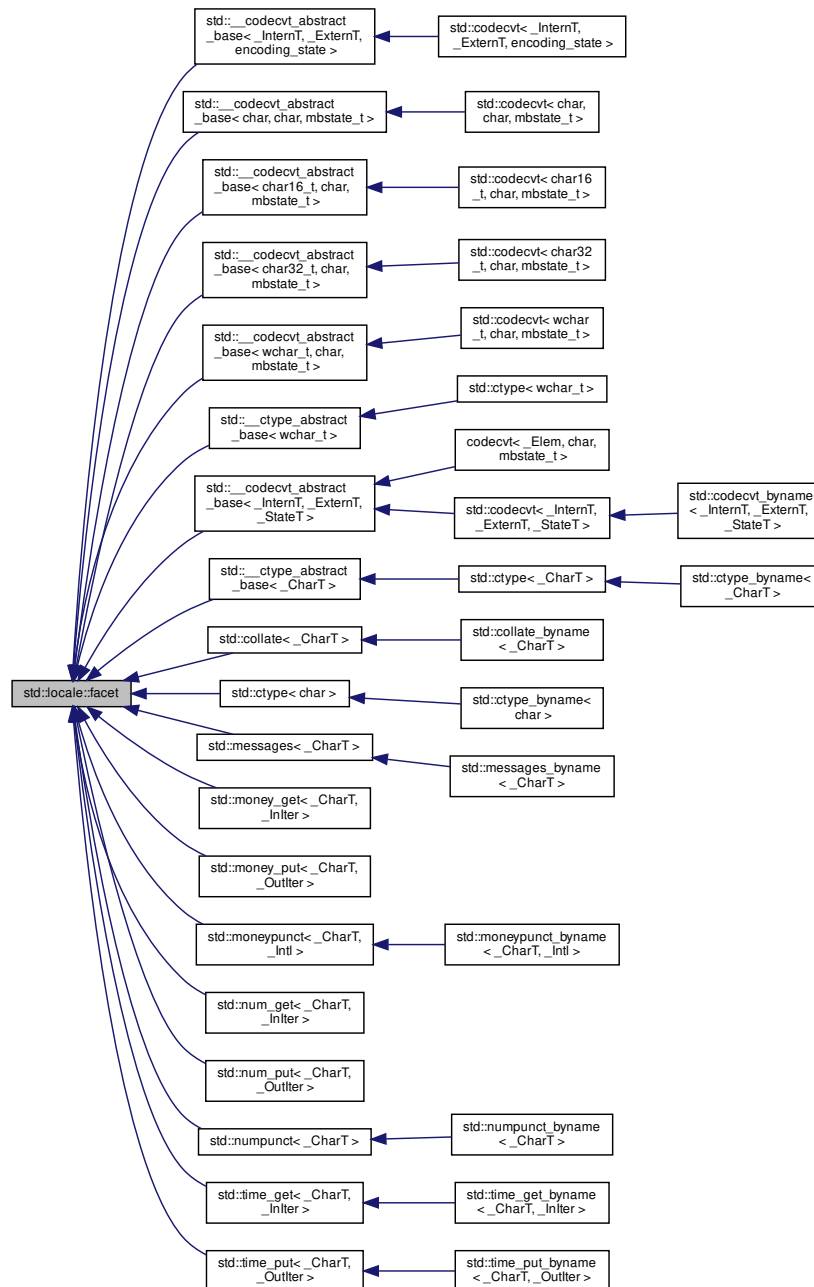
Definition at line 5214 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.483 std::locale::facet Class Reference

Inheritance diagram for std::locale::facet:



### Protected Member Functions

- [facet](#) (size\_t \_\_refs=0) throw ()

- **facet** (const facet &)=delete
- virtual ~facet ()
- **facet** & **operator=** (const facet &)=delete

#### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

#### Friends

- class **locale**
- class **locale::\_Impl**

#### 4.483.1 Detailed Description

Localization functionality base class.

The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.

Facets may not be copied or assigned.

Definition at line 373 of file locale\_classes.h.

#### 4.483.2 Constructor & Destructor Documentation

##### 4.483.2.1 facet()

```
std::locale::facet::facet (
 size_t __refs = 0) throw () [inline], [explicit], [protected]
```

Facet constructor.

This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

#### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__refs</code> | The initial value for reference count. |
|---------------------|----------------------------------------|

Definition at line 405 of file locale\_classes.h.

#### 4.483.2.2 ~facet()

```
virtual std::locale::facet::~~facet () [protected], [virtual]
```

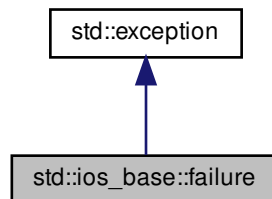
Facet destructor.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

### 4.484 std::ios\_base::failure Class Reference

Inheritance diagram for std::ios\_base::failure:



#### Public Member Functions

- **failure** (const [string](#) &\_\_str) throw ()
- **failure** (const [string](#) &\_\_s, const [error\\_code](#) &) noexcept
- **failure** (const char \*\_\_s, const [error\\_code](#) &=error\_code{})
- [error\\_code](#) **code** () const noexcept
- virtual const char \* [what](#) () const throw ()

#### 4.484.1 Detailed Description

These are thrown to indicate problems with io.

27.4.2.1.1 Class ios\_base::failure.

Definition at line 276 of file ios\_base.h.

## 4.484.2 Member Function Documentation

## 4.484.2.1 what()

```
virtual const char* std::ios_base::failure::what () const throw () [virtual]
```

Returns a C-style character string describing the general cause of the current error.

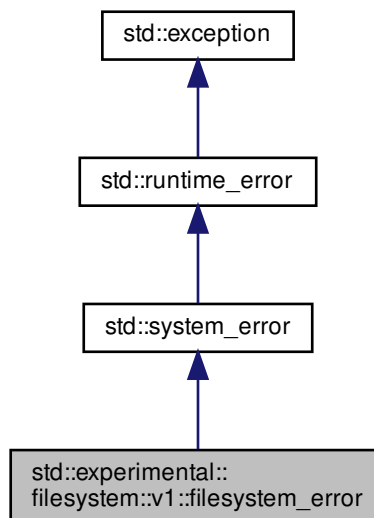
Reimplemented from [std::exception](#).

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

## 4.485 std::experimental::filesystem::v1::filesystem\_error Class Reference

Inheritance diagram for std::experimental::filesystem::v1::filesystem\_error:



## Public Member Functions

- **filesystem\_error** (const [string](#) &\_\_what\_arg, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, [error\\_code](#) \_\_ec)
- **filesystem\_error** (const [string](#) &\_\_what\_arg, const [path](#) &\_\_p1, const [path](#) &\_\_p2, [error\\_code](#) \_\_ec)
- const [error\\_code](#) & **code** () const noexcept
- const [path](#) & **path1** () const noexcept
- const [path](#) & **path2** () const noexcept
- const char \* **what** () const noexcept

#### 4.485.1 Detailed Description

Exception type thrown by the Filesystem TS library.

Definition at line 696 of file `experimental/bits/fs_path.h`.

#### 4.485.2 Member Function Documentation

##### 4.485.2.1 `what()`

```
const char* std::experimental::filesystem::v1::filesystem_error::what () const [inline], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::runtime\\_error](#).

Definition at line 715 of file `experimental/bits/fs_path.h`.

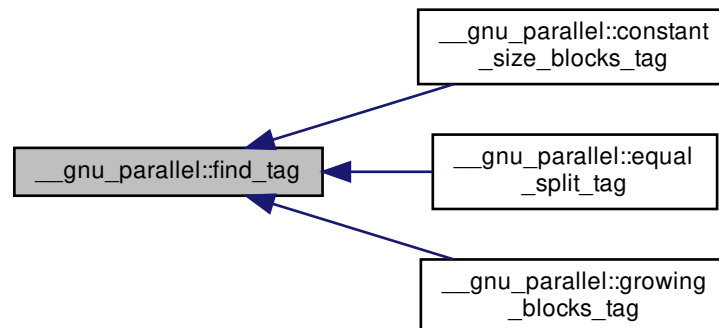
References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

#### 4.486 `__gnu_parallel::find_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::find_tag`:



## 4.486.1 Detailed Description

Base class for for std::find() variants.

Definition at line 104 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.487 std::fisher\_f\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- **fisher\_f\_distribution** (\_RealType \_\_m, \_RealType \_\_n=\_RealType(1))
- **fisher\_f\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **m** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- \_RealType **n** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::fisher_f_distribution< _RealType1 > &__x)`
- `bool operator== (const fisher_f_distribution &__d1, const fisher_f_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::fisher_f_distribution< _RealType1 > &__x)`

### 4.487.1 Detailed Description

```
template<typename _RealType = double>
class std::fisher_f_distribution< _RealType >
```

A fisher\_f\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)} \left(\frac{m}{n}\right)^{m/2} x^{(m/2)-1} \left(1 + \frac{mx}{n}\right)^{-(m+n)/2}$$

Definition at line 3062 of file random.h.

### 4.487.2 Member Typedef Documentation

#### 4.487.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::fisher_f_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 3065 of file random.h.

### 4.487.3 Member Function Documentation

#### 4.487.3.1 m()

```
template<typename _RealType = double>
_RealType std::fisher_f_distribution< _RealType >::m () const [inline]
```

Definition at line 3131 of file random.h.

Referenced by `std::fisher_f_distribution< _RealType >::operator()()`.



#### 4.487.3.2 max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution<_RealType>::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3164 of file random.h.

References std::numeric\_limits<\_Tp>::max().

#### 4.487.3.3 min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution<_RealType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3157 of file random.h.

#### 4.487.3.4 operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::fisher_f_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 3172 of file random.h.

References std::fisher\_f\_distribution<\_RealType>::m().

#### 4.487.3.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3142 of file random.h.

#### 4.487.3.6 param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution<_RealType>::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3150 of file random.h.

#### 4.487.3.7 reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 3121 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

### 4.487.4 Friends And Related Function Documentation

#### 4.487.4.1 operator<<

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::fisher_f_distribution< _RealType1 > & __x) [friend]
```

Inserts a `fisher_f_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                |
| <code>__x</code>  | A <code>fisher_f_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### 4.487.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
```

```
const fisher_f_distribution<_RealType> & __d1,
const fisher_f_distribution<_RealType> & __d2) [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.



Definition at line 3220 of file random.h.

#### 4.487.4.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits> & __is,
 std::fisher_f_distribution<_RealType1> & __x) [friend]
```

Extracts a fisher\_f\_distribution random number distribution \_\_x from the input stream \_\_is.

##### Parameters

|                                                                                            |                                                         |
|--------------------------------------------------------------------------------------------|---------------------------------------------------------|
| <br>__is  | An input stream.                                        |
| <br>__x | A fisher_f_distribution random number generator engine. |

##### Returns

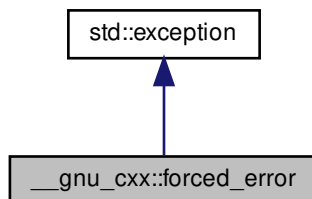
The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.488 `__gnu_cxx::forced_error` Struct Reference

Inheritance diagram for `__gnu_cxx::forced_error`:



##### Public Member Functions

- virtual const char \* [what](#) () const noexcept

##### 4.488.1 Detailed Description

Thrown by exception safety machinery.

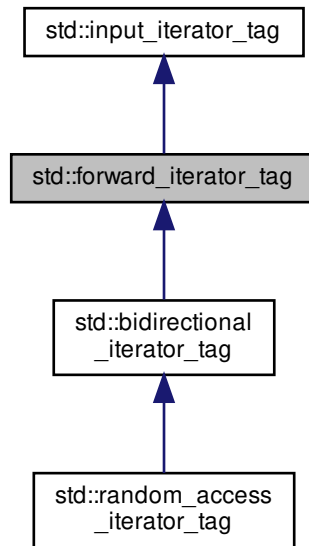
Definition at line 79 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.489 std::forward\_iterator\_tag Struct Reference

Inheritance diagram for std::forward\_iterator\_tag:



## 4.489.1 Detailed Description

Forward iterators support a superset of input iterator operations.

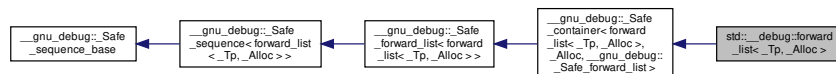
Definition at line 99 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.490 std::\_\_debug::forward\_list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::forward\_list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Base::allocator_type` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, forward_list>` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, forward_list>` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **forward\_list** (const allocator\_type &\_\_al) noexcept
- **forward\_list** (const forward\_list &\_\_list, const allocator\_type &\_\_al)
- **forward\_list** (forward\_list &&\_\_list, const allocator\_type &\_\_al)
- **forward\_list** (size\_type \_\_n, const allocator\_type &\_\_al=allocator\_type())
- **forward\_list** (size\_type \_\_n, const\_Tp &\_\_value, const allocator\_type &\_\_al=allocator\_type())
- template<typename \_\_InputIterator, typename = std::::RequireInputIter<\_\_InputIterator>>>  
**forward\_list** (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last, const allocator\_type &\_\_al=allocator\_type())
- **forward\_list** (const forward\_list &)=default
- **forward\_list** (forward\_list &&)=default
- **forward\_list** (std::initializer\_list<\_Tp> > \_\_il, const allocator\_type &\_\_al=allocator\_type())
- `_Base & _M_base` () noexcept
- const `_Base & _M_base` () const noexcept
- void `_M_invalidate_if` (\_\_Predicate \_\_pred)
- void `_M_swap` (\_\_Safe\_container &\_\_x) noexcept
- void `_M_transfer_from_if` (\_\_Safe\_sequence &\_\_from, \_\_Predicate \_\_pred)
- template<typename \_\_InputIterator, typename = std::::RequireInputIter<\_\_InputIterator>>>  
void **assign** (\_\_InputIterator \_\_first, \_\_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const\_Tp &\_\_val)
- void **assign** (std::initializer\_list<\_Tp> > \_\_il)
- **iterator before\_begin** () noexcept
- **const\_iterator before\_begin** () const noexcept
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbefore\_begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- template<typename... \_\_Args>  
**iterator emplace\_after** (const\_iterator \_\_pos, \_\_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **iterator erase\_after** (const\_iterator \_\_pos)
- **iterator erase\_after** (const\_iterator \_\_pos, const\_iterator \_\_last)
- reference **front** ()
- const\_reference **front** () const

- iterator **insert\_after** (const\_iterator \_\_pos, const \_Tp &\_\_val)
- iterator **insert\_after** (const\_iterator \_\_pos, \_Tp &&\_\_val)
- iterator **insert\_after** (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
- template<typename \_InputIterator, typename = std::\_\_RequireInputIter<\_InputIterator>>  
iterator **insert\_after** (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert\_after** (const\_iterator \_\_pos, std::initializer\_list<\_Tp> \_\_il)
- void **merge** (forward\_list &&\_\_list)
- void **merge** (forward\_list &\_\_list)
- template<typename \_Comp >  
void **merge** (forward\_list &&\_\_list, \_Comp \_\_comp)
- template<typename \_Comp >  
void **merge** (forward\_list &\_\_list, \_Comp \_\_comp)
- forward\_list & **operator=** (const forward\_list &)=default
- forward\_list & **operator=** (forward\_list &&)=default
- forward\_list & **operator=** (std::initializer\_list<\_Tp> \_\_il)
- void **pop\_front** ()
- \_\_remove\_return\_type **remove** (const \_Tp &\_\_val)
- template<typename \_Pred >  
\_\_remove\_return\_type **remove\_if** (\_Pred \_\_pred)
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const value\_type &\_\_val)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **splice\_after** (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_before, const\_iterator \_\_last)
- void **swap** (forward\_list &\_\_list) noexcept(noexcept(declval<\_Base &>().swap(\_\_list)))
- \_\_remove\_return\_type **unique** ()
- template<typename \_BinPred >  
\_\_remove\_return\_type **unique** (\_BinPred \_\_binary\_pred)

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_invalidate\_all** ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_revalidate\_singular** ()
- \_Safe\_container & **\_M\_safe** () noexcept
- void **\_M\_swap** (\_Safe\_sequence\_base &) noexcept

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
`class ::__gnu_debug::_Safe_iterator`

### 4.490.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list<_Tp, _Alloc >
```

Class `std::forward_list` with safety/checking/debug instrumentation.

Definition at line 36 of file `debug/forward_list`.

### 4.490.2 Member Function Documentation

#### 4.490.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~_Safe_sequence_base()`.

#### 4.490.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 4.490.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.



## 4.490.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 4.490.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

## 4.490.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.490.2.7 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< forward_list< _Tp, _Alloc > >::_M_transfer_from_if (
 _Safe_sequence< forward_list< _Tp, _Alloc > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

## 4.490.3 Member Data Documentation

#### 4.490.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.490.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.490.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

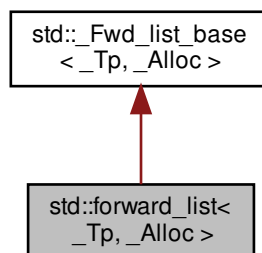
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/forward\\_list](#)

### 4.491 `std::forward_list< _Tp, _Alloc >` Class Template Reference

Inheritance diagram for `std::forward_list< _Tp, _Alloc >`:



## Public Types

- typedef `_Alloc allocator_type`
- typedef `_Base::const_iterator const_iterator`
- typedef `_Alloc_traits::const_pointer const_pointer`
- typedef `const value_type & const_reference`
- typedef `std::ptrdiff_t difference_type`
- typedef `_Base::iterator iterator`
- typedef `_Alloc_traits::pointer pointer`
- typedef `value_type & reference`
- typedef `std::size_t size_type`
- typedef `_Tp value_type`

## Public Member Functions

- `forward_list()` = default
- `forward_list(const _Alloc &__al)` noexcept
- `forward_list(const forward_list &__list, const _Alloc &__al)`
- `forward_list(forward_list &&__list, const _Alloc &__al)` noexcept(`_Node_alloc_traits::_S_always_equal()`)
- `forward_list(size_type __n, const _Alloc &__al= _Alloc())`
- `forward_list(size_type __n, const _Tp &__value, const _Alloc &__al= _Alloc())`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`forward_list(_InputIterator __first, _InputIterator __last, const _Alloc &__al= _Alloc())`
- `forward_list(const forward_list &__list)`
- `forward_list(forward_list &&)=default`
- `forward_list(std::initializer_list<_Tp> __il, const _Alloc &__al= _Alloc())`
- `~forward_list()` noexcept
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>`  
`void assign(_InputIterator __first, _InputIterator __last)`
- `void assign(size_type __n, const _Tp &__val)`
- `void assign(std::initializer_list<_Tp> __il)`
- `iterator before_begin()` noexcept
- `const_iterator before_begin()` const noexcept
- `iterator begin()` noexcept
- `const_iterator begin()` const noexcept
- `const_iterator cbefore_begin()` const noexcept
- `const_iterator cbegin()` const noexcept
- `const_iterator cend()` const noexcept
- `void clear()` noexcept
- `template<typename... _Args>`  
`iterator emplace_after(const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_front(_Args &&... __args)`
- `bool empty()` const noexcept
- `iterator end()` noexcept
- `const_iterator end()` const noexcept
- `iterator erase_after(const_iterator __pos)`
- `iterator erase_after(const_iterator __pos, const_iterator __last)`
- `reference front()`
- `const_reference front()` const

- allocator\_type `get_allocator` () const noexcept
  - iterator `insert_after` (const\_iterator \_\_pos, const \_Tp &\_\_val)
  - iterator `insert_after` (const\_iterator \_\_pos, \_Tp &&\_\_val)
  - iterator `insert_after` (const\_iterator \_\_pos, size\_type \_\_n, const \_Tp &\_\_val)
  - template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator `insert_after` (const\_iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last)
  - iterator `insert_after` (const\_iterator \_\_pos, std::initializer\_list<\_Tp> \_\_il)
  - size\_type `max_size` () const noexcept
  - void `merge` (forward\_list &&\_\_list)
  - void `merge` (forward\_list &\_\_list)
  - template<typename \_Comp>  
void `merge` (forward\_list &&\_\_list, \_Comp \_\_comp)
  - template<typename \_Comp>  
void `merge` (forward\_list &\_\_list, \_Comp \_\_comp)
  - forward\_list & `operator=` (const forward\_list &\_\_list)
  - forward\_list & `operator=` (forward\_list &&\_\_list) noexcept(\_Node\_alloc\_traits::\_S\_nothrow\_move())
  - forward\_list & `operator=` (std::initializer\_list<\_Tp> \_\_il)
  - void `pop_front` ()
  - void `push_front` (const \_Tp &\_\_val)
  - void `push_front` (\_Tp &&\_\_val)
  - \_\_remove\_return\_type `remove` (const \_Tp &\_\_val)
  - template<typename \_Pred>  
auto `remove_if` (\_Pred \_\_pred) -> \_\_remove\_return\_type
  - template<typename \_Pred>  
\_\_remove\_return\_type `remove_if` (\_Pred \_\_pred)
  - void `resize` (size\_type \_\_sz)
  - void `resize` (size\_type \_\_sz, const value\_type &\_\_val)
  - void `reverse` () noexcept
  - void `sort` ()
  - template<typename \_Comp>  
void `sort` (\_Comp \_\_comp)
  - void `splice_after` (const\_iterator \_\_pos, forward\_list &&\_\_list) noexcept
  - void `splice_after` (const\_iterator \_\_pos, forward\_list &\_\_list) noexcept
  - void `splice_after` (const\_iterator \_\_pos, forward\_list &&\_\_list, const\_iterator \_\_i) noexcept
  - void `splice_after` (const\_iterator \_\_pos, forward\_list &\_\_list, const\_iterator \_\_i) noexcept
  - void `swap` (forward\_list &\_\_list) noexcept
  - template<typename \_BinPred>  
auto `unique` (\_BinPred \_\_binary\_pred) -> \_\_remove\_return\_type
  - \_\_remove\_return\_type `unique` ()
  - template<typename \_BinPred>  
\_\_remove\_return\_type `unique` (\_BinPred \_\_binary\_pred)
- 
- void `splice_after` (const\_iterator \_\_pos, forward\_list &&, const\_iterator \_\_before, const\_iterator \_\_last) noexcept
  - void `splice_after` (const\_iterator \_\_pos, forward\_list &, const\_iterator \_\_before, const\_iterator \_\_last) noexcept

## Private Member Functions

- `template<typename... _Args>`  
`_Node * _M_create_node ( _Args &&... __args)`
- `_Fwd_list_node_base * _M_erase_after ( _Fwd_list_node_base * __pos)`
- `_Fwd_list_node_base * _M_erase_after ( _Fwd_list_node_base * __pos, _Fwd_list_node_base * __last)`
- `_Node * _M_get_node ()`
- `_Node_alloc_type & _M_get_Node_allocator () noexcept`
- `const _Node_alloc_type & _M_get_Node_allocator () const noexcept`
- `template<typename... _Args>`  
`_Fwd_list_node_base * _M_insert_after (const_iterator __pos, _Args &&... __args)`
- `void _M_put_node ( _Node * __p)`

## Private Attributes

- `_Fwd_list_impl _M_impl`

## 4.491.1 Detailed Description

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list< _Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

## Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`, random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::forward_list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

Definition at line 423 of file `forward_list.h`.

## 4.491.2 Constructor &amp; Destructor Documentation

**4.491.2.1 forward\_list()** [1/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list () [default]
```

Creates a forward\_list with no elements.

**4.491.2.2 forward\_list()** [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 const _Alloc & __al) [inline], [explicit], [noexcept]
```

Creates a forward\_list with no elements.

**Parameters**

|                   |                      |
|-------------------|----------------------|
| <code>__al</code> | An allocator object. |
|-------------------|----------------------|

Definition at line 466 of file forward\_list.h.

**4.491.2.3 forward\_list()** [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 const forward_list< _Tp, _Alloc > & __list,
 const _Alloc & __al) [inline]
```

Copy constructor with allocator argument.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to copy.  |
| <code>__al</code>   | An allocator object. |

Definition at line 475 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::begin()`, and `std::forward_list< _Tp, _Alloc >::end()`.

**4.491.2.4 forward\_list()** [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
```

```
forward_list<_Tp, _Alloc > && __list,
const _Alloc & __al) [inline], [noexcept]
```

Move constructor with allocator argument.

#### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__list</code> | Input list to move.  |
| <code>__al</code>   | An allocator object. |

Definition at line 503 of file forward\_list.h.

#### 4.491.2.5 forward\_list() [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc >::forward_list (
 size_type __n,
 const _Alloc & __al = _Alloc()) [inline], [explicit]
```

Creates a forward\_list with default constructed elements.

#### Parameters

|                   |                                             |
|-------------------|---------------------------------------------|
| <code>__n</code>  | The number of elements to initially create. |
| <code>__al</code> | An allocator object.                        |

This constructor creates the forward\_list with `__n` default constructed elements.

Definition at line 518 of file forward\_list.h.

#### 4.491.2.6 forward\_list() [6/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc >::forward_list (
 size_type __n,
 const _Tp & __value,
 const _Alloc & __al = _Alloc()) [inline]
```

Creates a forward\_list with copies of an exemplar element.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__al</code>    | An allocator object.                        |

This constructor fills the `forward_list` with `__n` copies of `__value`.

Definition at line 531 of file `forward_list.h`.

#### 4.491.2.7 `forward_list()` [7/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::forward_list<_Tp, _Alloc>::forward_list (
 _InputIterator __first,
 _InputIterator __last,
 const _Alloc & __al = _Alloc()) [inline]
```

Builds a `forward_list` from a range.

##### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__al</code>    | An allocator object. |

Create a `forward_list` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is distance(`↵`  
`__first,__last`)).

Definition at line 548 of file `forward_list.h`.

#### 4.491.2.8 `forward_list()` [8/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list<_Tp, _Alloc>::forward_list (
 const forward_list<_Tp, _Alloc> & __list) [inline]
```

The `forward_list` copy constructor.

##### Parameters

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of identical element and allocator types. |
|---------------------|-----------------------------------------------------------------------|

Definition at line 558 of file `forward_list.h`.

References `std::forward_list<_Tp, _Alloc>::begin()`, and `std::forward_list<_Tp, _Alloc>::end()`.



## 4.491.2.9 forward\_list() [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 forward_list< _Tp, _Alloc > &&) [default]
```

The forward\_list move constructor.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The newly-created forward\_list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified forward\_list.

## 4.491.2.10 forward\_list() [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
 std::initializer_list< _Tp > __il,
 const _Alloc & __al = _Alloc()) [inline]
```

Builds a forward\_list from an initializer\_list.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
| <code>__al</code> | An allocator object.               |

Create a forward\_list consisting of copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

Definition at line 582 of file forward\_list.h.

## 4.491.2.11 ~forward\_list()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~~forward_list () [inline], [noexcept]
```

The forward\_list dtor.

Definition at line 590 of file forward\_list.h.

## 4.491.3 Member Function Documentation

**4.491.3.1 assign()** [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>>
void std::forward_list<_Tp, _Alloc>::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a forward\_list.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a forward\_list with copies of the elements in the range [`__first`,`__last`).

Note that the assignment completely changes the forward\_list and that the number of elements of the resulting forward\_list is the same as the number of elements assigned.

Definition at line 658 of file forward\_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::assign()`, and `std::forward_list<_Tp, _Alloc>::operator=()`.

**4.491.3.2 assign()** [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::assign (
 size_type __n,
 const _Tp & __val) [inline]
```

Assigns a given value to a forward\_list.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a forward\_list with `__n` copies of the given value. Note that the assignment completely changes the forward\_list, and that the resulting forward\_list has `__n` elements.


Definition at line 675 of file forward\_list.h.

## 4.491.3.3 assign() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
 std::initializer_list< _Tp > __il) [inline]
```

Assigns an initializer\_list to a forward\_list.

## Parameters

|                                                                                   |                                    |
|-----------------------------------------------------------------------------------|------------------------------------|
|  | An initializer_list of value_type. |
|-----------------------------------------------------------------------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list \_\_il. This is linear in il.size().

Definition at line 687 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::assign().

## 4.491.3.4 before\_begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::before_begin () [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 702 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::insert\_after().

## 4.491.3.5 before\_begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::before_begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 711 of file forward\_list.h.

**4.491.3.6 begin()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 719 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::forward_list()`.

**4.491.3.7 begin()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 728 of file `forward_list.h`.

**4.491.3.8 cbefore\_begin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 764 of file `forward_list.h`.

Referenced by `std::forward_list< _Tp, _Alloc >::emplace_front()`, and `std::forward_list< _Tp, _Alloc >::push_front()`.

**4.491.3.9 cbegin()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 755 of file `forward_list.h`.

Referenced by `std::operator<()`, `std::forward_list< _Tp, _Alloc >::operator=()`, and `std::operator==(())`.

## 4.491.3.10 cend()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 773 of file forward\_list.h.

Referenced by std::operator<(), std::forward\_list< \_Tp, \_Alloc >::operator=(), and std::operator==( ).

## 4.491.3.11 clear()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::clear () [inline], [noexcept]
```

Erases all the elements.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1086 of file forward\_list.h.

## 4.491.3.12 emplace\_after()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list< _Tp, _Alloc >::emplace_after (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Constructs object in forward\_list after the specified iterator.

## Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>__pos</code>  | A const_iterator into the forward_list. |
| <code>__args</code> | Arguments.                              |

## Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 895 of file forward\_list.h.

#### 4.491.3.13 `emplace_front()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
void std::forward_list<_Tp, _Alloc>::emplace_front (
 _Args &&... __args) [inline]
```

Constructs object in forward\_list at the front of the list.

##### Parameters

|                     |            |
|---------------------|------------|
| <code>__args</code> | Arguments. |
|---------------------|------------|

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...)`  at the front of the list. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 834 of file forward\_list.h.

References `std::forward_list<_Tp, _Alloc>::cbefore_begin()`, and `std::forward_list<_Tp, _Alloc>::front()`.

#### 4.491.3.14 `empty()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list<_Tp, _Alloc>::empty () const [inline], [noexcept]
```

Returns true if the `forward_list` is empty. (Thus `begin()` would equal `end()`.)

Definition at line 781 of file forward\_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::insert_after()`.

#### 4.491.3.15 `end()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list<_Tp, _Alloc>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `forward_list`. Iteration is done in ordinary element order.

Definition at line 737 of file forward\_list.h.

Referenced by `std::forward_list<_Tp, _Alloc>::forward_list()`, and `std::forward_list<_Tp, _Alloc>::insert_after()`.

**4.491.3.16** end() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only iterator that points one past the last element in the forward\_list. Iteration is done in ordinary element order.

Definition at line 746 of file forward\_list.h.

**4.491.3.17** erase\_after() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
 const_iterator __pos) [inline]
```

Removes the element pointed to by the iterator following pos.

**Parameters**

|                    |                                                |
|--------------------|------------------------------------------------|
| <code>__pos</code> | Iterator pointing before element to be erased. |
|--------------------|------------------------------------------------|

**Returns**

An iterator pointing to the element following the one that was erased, or end() if no such element exists.

This function will erase the element at the given position and thus shorten the forward\_list by one.

Due to the nature of a forward\_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 998 of file forward\_list.h.

**4.491.3.18** erase\_after() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
 const_iterator __pos,
 const_iterator __last) [inline]
```

Remove a range of elements.

**Parameters**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator pointing before the first element to be erased.     |
| <code>__last</code> | Iterator pointing to one past the last element to be erased. |

**Returns**

@ \_\_last.

This function will erase the elements in the range ( \_\_pos, \_\_last) and shorten the forward\_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1021 of file forward\_list.h.

**4.491.3.19 front()** [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list< _Tp, _Alloc >::front () [inline]
```

Returns a read/write reference to the data at the first element of the forward\_list.

Definition at line 798 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::emplace\_front().

**4.491.3.20 front()** [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list< _Tp, _Alloc >::front () const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the forward\_list.

Definition at line 809 of file forward\_list.h.

**4.491.3.21 get\_allocator()**

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 692 of file forward\_list.h.

**4.491.3.22 insert\_after()** [1/5]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
 const_iterator __pos,
 const _Tp & __val) [inline]
```

Inserts given value into forward\_list after specified iterator.



## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__pos</code> | An iterator into the forward_list. |
| <code>__val</code> | Data to be inserted.               |

## Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward\_list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 912 of file forward\_list.h.

Referenced by std::forward\_list< \_Tp, \_Alloc >::insert\_after().

## 4.491.3.23 insert\_after() [2/5]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
 const_iterator __pos,
 _Tp && __val) [inline]
```

Definition at line 919 of file forward\_list.h.

References std::move().

## 4.491.3.24 insert\_after() [3/5]

```
template<typename _Tp, typename _Alloc >
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
 const_iterator __pos,
 size_type __n,
 const _Tp & __val)
```

Inserts a number of copies of given data into the forward\_list.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__pos</code> | An iterator into the forward_list. |
| <code>__n</code>   | Number of elements to be inserted. |
| <code>__val</code> | Data to be inserted.               |

**Returns**

An iterator pointing to the last inserted copy of *val* or *pos* if *n* == 0.

This function will insert a specified number of copies of the given data after the location specified by *pos*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 256 of file `forward_list.tcc`.

References `std::forward_list< _Tp, _Alloc >::before_begin()`, and `std::forward_list< _Tp, _Alloc >::end()`.

**4.491.3.25 insert\_after()** [4/5]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator , typename >
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
 const_iterator __pos,
 _InputIterator __first,
 _InputIterator __last)
```

Inserts a range into the `forward_list`.

**Parameters**

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <code>__pos</code>   | An iterator into the <code>forward_list</code> . |
| <code>__first</code> | An input iterator.                               |
| <code>__last</code>  | An input iterator.                               |

**Returns**

An iterator pointing to the last inserted element or `__pos` if `__first == __last`.

This function will insert copies of the data in the range `[__first,__last)` into the `forward_list` after the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 271 of file `forward_list.tcc`.

References `std::forward_list< _Tp, _Alloc >::before_begin()`, `std::forward_list< _Tp, _Alloc >::empty()`, and `std::forward_list< _Tp, _Alloc >::end()`.

**4.491.3.26 insert\_after()** [5/5]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
 const_iterator __pos,
 std::initializer_list< _Tp > __il) [inline]
```

Inserts the contents of an `initializer_list` into `forward_list` after the specified iterator.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__pos</code> | An iterator into the forward_list. |
| <code>__il</code>  | An initializer_list of value_type. |

## Returns

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the initializer\_list `__il` into the forward\_list before the location specified by `__pos`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 977 of file forward\_list.h.

References `std::forward_list< _Tp, _Alloc >::insert_after()`.

## 4.491.3.27 max\_size()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the largest possible number of elements of forward\_list.

Definition at line 788 of file forward\_list.h.

References `std::allocator_traits< _Alloc >::max_size()`.

## 4.491.3.28 merge() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::merge (
 forward_list< _Tp, _Alloc > && __list) [inline]
```

Merge sorted lists.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__list</code> | Sorted list to merge. |
|---------------------|-----------------------|

Assumes that both `list` and this list are sorted according to operator<(). Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equal.

Definition at line 1242 of file forward\_list.h.

References `std::move()`.

#### 4.491.3.29 `merge()` [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _Comp >
void forward_list::merge (
 forward_list< _Tp, _Alloc > && __list,
 _Comp __comp)
```

Merge sorted lists according to comparison function.

##### Parameters

|                     |                                          |
|---------------------|------------------------------------------|
| <code>__list</code> | Sorted list to merge.                    |
| <code>__comp</code> | Comparison function defining sort order. |

Assumes that both `__list` and this list are sorted according to `comp`. Merges elements of `__list` into this list in sorted order, leaving `__list` empty when complete. Elements in this list precede elements in `__list` that are equivalent according to `comp()`.

Definition at line 373 of file `forward_list.tcc`.

References `std::move()`.

#### 4.491.3.30 `operator=()` [1/3]

```
template<typename _Tp , typename _Alloc >
forward_list< _Tp, _Alloc > & forward_list::operator= (
 const forward_list< _Tp, _Alloc > & __list)
```

The `forward_list` assignment operator.

##### Parameters

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of identical element and allocator types. |
|---------------------|-----------------------------------------------------------------------|

All the elements of `__list` are copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 140 of file `forward_list.tcc`.

References `std::__addressof()`, `std::forward_list< _Tp, _Alloc >::cbegin()`, and `std::forward_list< _Tp, _Alloc >::cend()`.

## 4.491.3.31 operator=() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
 forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

The forward\_list move assignment operator.

## Parameters

|                     |                                                          |
|---------------------|----------------------------------------------------------|
| <code>__list</code> | A forward_list of identical element and allocator types. |
|---------------------|----------------------------------------------------------|

The contents of `__list` are moved into this forward\_list (without copying, if the allocators permit it).

Afterwards `__list` is a valid, but unspecified forward\_list

Whether the allocator is moved depends on the allocator traits.

Definition at line 618 of file forward\_list.h.

References std::move().

## 4.491.3.32 operator=() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list& std::forward_list< _Tp, _Alloc >::operator= (
 std::initializer_list< _Tp > __il) [inline]
```

The forward\_list initializer list assignment operator.

## Parameters

|                   |                                    |
|-------------------|------------------------------------|
| <code>__il</code> | An initializer_list of value_type. |
|-------------------|------------------------------------|

Replace the contents of the forward\_list with copies of the elements in the initializer\_list `__il`. This is linear in `__il.size()`.

Definition at line 637 of file forward\_list.h.

References std::forward\_list< \_Tp, \_Alloc >::assign().

## 4.491.3.33 pop\_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the `forward_list` by one. Due to the nature of a `forward_list` this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

Definition at line 877 of file `forward_list.h`.

#### 4.491.3.34 `push_front()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
 const _Tp & __val) [inline]
```

Add data to the front of the `forward_list`.

##### Parameters

|                    |                   |
|--------------------|-------------------|
| <code>__val</code> | Data to be added. |
|--------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the `forward_list` and assigns the given data to it. Due to the nature of a `forward_list` this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 854 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`.

#### 4.491.3.35 `push_front()` [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
 _Tp && __val) [inline]
```

Definition at line 861 of file `forward_list.h`.

References `std::forward_list< _Tp, _Alloc >::cbefore_begin()`, and `std::move()`.

#### 4.491.3.36 `remove()`

```
template<typename _Tp, typename _Alloc >
auto forward_list::remove (
 const _Tp & __val)
```

Remove all elements equal to value.

## Parameters

|                    |                      |
|--------------------|----------------------|
| <code>__val</code> | The value to remove. |
|--------------------|----------------------|

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 290 of file `forward_list.tcc`.

References `std::__addressof()`.

## 4.491.3.37 remove\_if()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _Pred >
__remove_return_type std::forward_list< _Tp, _Alloc >::remove_if (
 _Pred __pred)
```

Remove all elements satisfying a predicate.

## Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__pred</code> | Unary predicate function or object. |
|---------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 4.491.3.38 resize() [1/2]

```
template<typename _Tp , typename _Alloc >
void forward_list::resize (
 size_type __sz)
```

Resizes the `forward_list` to the specified number of elements.

## Parameters

|                   |                                                                  |
|-------------------|------------------------------------------------------------------|
| <code>__sz</code> | Number of elements the <code>forward_list</code> should contain. |
|-------------------|------------------------------------------------------------------|

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and the new elements are default constructed.

Definition at line 182 of file `forward_list.tcc`.

References `std::initializer_list<_E>::end()`.

#### 4.491.3.39 `resize()` [2/2]

```
template<typename _Tp, typename _Alloc >
void forward_list::resize (
 size_type __sz,
 const value_type & __val)
```

Resizes the `forward_list` to the specified number of elements.

##### Parameters

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| <code>__sz</code>  | Number of elements the <code>forward_list</code> should contain. |
| <code>__val</code> | Data with which new elements should be populated.                |

This function will resize the `forward_list` to the specified number of elements. If the number is smaller than the `forward_list`'s current number of elements the `forward_list` is truncated, otherwise the `forward_list` is extended and new elements are populated with given data.

Definition at line 201 of file `forward_list.tcc`.

References `std::initializer_list<_E>::end()`.

#### 4.491.3.40 `reverse()`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::reverse () [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1295 of file `forward_list.h`.

#### 4.491.3.41 `sort()` [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list<_Tp, _Alloc>::sort () [inline]
```

Sort the elements of the list.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 1276 of file `forward_list.h`.



## 4.491.3.42 sort() [2/2]

```
template<typename _Tp , class _Alloc >
template<typename _Comp >
void forward_list::sort (
 _Comp __comp)
```

Sort the forward\_list using a comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 417 of file forward\_list.tcc.

## 4.491.3.43 splice\_after() [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && __list) [inline], [noexcept]
```

Insert contents of another forward\_list.

## Parameters

|                     |                                                   |
|---------------------|---------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after. |
| <code>__list</code> | Source list.                                      |

The elements of *list* are inserted in constant time after the element referenced by *pos*. *list* becomes an empty list.

Requires this != x.

Definition at line 1103 of file forward\_list.h.

## 4.491.3.44 splice\_after() [2/4]

```
template<typename _Tp , typename _Alloc >
void forward_list::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && __list,
 const_iterator __i) [noexcept]
```

Insert element from another forward\_list.

## Parameters

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>__pos</code>  | Iterator referencing the element to insert after.            |
| <code>__list</code> | Source list.                                                 |
| <code>__i</code>    | Iterator referencing the element before the element to move. |

Removes the element in list *list* referenced by *i* and inserts it into the current list after *pos*.

Definition at line 239 of file forward\_list.tcc.

#### 4.491.3.45 splice\_after() [3/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > && ,
 const_iterator __before,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another forward\_list.

##### Parameters

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.       |
| <code>__list</code>   | Source list.                                            |
| <code>__before</code> | Iterator referencing before the start of range in list. |
| <code>__last</code>   | Iterator referencing the end of range in list.          |

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1147 of file forward\_list.h.

#### 4.491.3.46 splice\_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
 const_iterator __pos,
 forward_list< _Tp, _Alloc > & ,
 const_iterator __before,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another forward\_list.

##### Parameters

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <code>__pos</code>    | Iterator referencing the element to insert after.       |
| <code>__list</code>   | Source list.                                            |
| <code>__before</code> | Iterator referencing before the start of range in list. |
| <code>__last</code>   | Iterator referencing the end of range in list.          |

Removes elements in the range (`__before`,`__last`) and inserts them after `__pos` in constant time.

Undefined if `__pos` is in (`__before`,`__last`).

Definition at line 1152 of file `forward_list.h`.

#### 4.491.3.47 swap()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (
 forward_list< _Tp, _Alloc > & __list) [inline], [noexcept]
```

Swaps data with another `forward_list`.

##### Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__list</code> | A <code>forward_list</code> of the same element and allocator types. |
|---------------------|----------------------------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1040 of file `forward_list.h`.

#### 4.491.3.48 unique() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique () [inline]
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1211 of file `forward_list.h`.

#### 4.491.3.49 unique() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _BinPred >
__remove_return_type std::forward_list< _Tp, _Alloc >::unique (
 _BinPred __binary_pred)
```

Remove consecutive elements satisfying a predicate.

## Parameters

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>__binary_pred</code> | Binary predicate function or object. |
|----------------------------|--------------------------------------|

For each consecutive set of elements `[first,last)` that satisfy `predicate(first,i)` where `i` is an iterator in `[first,last)`, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

The documentation for this class was generated from the following files:

- [forward\\_list.h](#)
- [forward\\_list.tcc](#)

#### 4.492 `std::fpos<_StateT>` Class Template Reference

## Public Member Functions

- [fpos](#) ([streamoff](#) \_\_off)
- **fpos** (const [fpos](#) &)=default
- [operator streamoff](#) () const
- [fpos](#) [operator+](#) ([streamoff](#) \_\_off) const
- [fpos](#) & [operator+=](#) ([streamoff](#) \_\_off)
- [fpos](#) [operator-](#) ([streamoff](#) \_\_off) const
- [streamoff](#) [operator-](#) (const [fpos](#) &\_\_other) const
- [fpos](#) & [operator-=](#) ([streamoff](#) \_\_off)
- [fpos](#) & **operator=** (const [fpos](#) &)=default
- void [state](#) (\_StateT \_\_st)
- \_StateT [state](#) () const

##### 4.492.1 Detailed Description

```
template<typename _StateT>
class std::fpos<_StateT>
```

Class representing stream positions.

The standard places no requirements upon the template parameter `StateT`. In this implementation `StateT` must be `DefaultConstructible`, `CopyConstructible` and `Assignable`. The standard only requires that `fpos` should contain a member of type `StateT`. In this implementation it also contains an offset stored as a signed integer.

## Parameters

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>StateT</code> | Type passed to and returned from <code>state()</code> . |
|---------------------|---------------------------------------------------------|

Definition at line 112 of file postypes.h.

#### 4.492.2 Constructor & Destructor Documentation

##### 4.492.2.1 fpos()

```
template<typename _StateT>
std::fpos< _StateT >::fpos (
 streamoff __off) [inline]
```

Construct position from offset.

Definition at line 133 of file postypes.h.

#### 4.492.3 Member Function Documentation

##### 4.492.3.1 operator streamoff()

```
template<typename _StateT>
std::fpos< _StateT >::operator streamoff () const [inline]
```

Convert to streamoff.

Definition at line 143 of file postypes.h.

##### 4.492.3.2 operator+()

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator+ (
 streamoff __off) const [inline]
```

Add position and offset.

Definition at line 184 of file postypes.h.

#### 4.492.3.3 operator+=()

```
template<typename _StateT>
fpos& std::fpos< _StateT >::operator+= (
 streamoff __off) [inline]
```

Add offset to this position.

Definition at line 160 of file postypes.h.

#### 4.492.3.4 operator-() [1/2]

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator- (
 streamoff __off) const [inline]
```

Subtract offset from position.

Definition at line 198 of file postypes.h.

#### 4.492.3.5 operator-() [2/2]

```
template<typename _StateT>
streamoff std::fpos< _StateT >::operator- (
 const fpos< _StateT > & __other) const [inline]
```

Subtract position to return offset.

Definition at line 211 of file postypes.h.

#### 4.492.3.6 operator-=()

```
template<typename _StateT>
fpos& std::fpos< _StateT >::operator-= (
 streamoff __off) [inline]
```

Subtract offset from this position.

Definition at line 171 of file postypes.h.

## 4.492.3.7 state() [1/2]

```
template<typename _StateT>
void std::fpos< _StateT >::state (
 _StateT __st) [inline]
```

Remember the value of *st*.

Definition at line 147 of file postypes.h.

## 4.492.3.8 state() [2/2]

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state () const [inline]
```

Return the last set value of *st*.

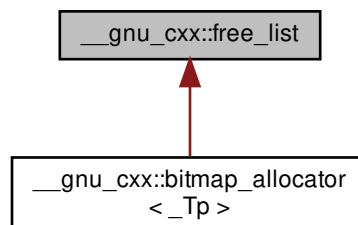
Definition at line 152 of file postypes.h.

The documentation for this class was generated from the following file:

- [postypes.h](#)

## 4.493 \_\_gnu\_cxx::free\_list Class Reference

Inheritance diagram for \_\_gnu\_cxx::free\_list:



## Public Types

- typedef \_\_mutex **\_\_mutex\_type**
- typedef vector\_type::iterator **iterator**
- typedef std::size\_t \* **value\_type**
- typedef [\\_\\_detail::\\_\\_mini\\_vector](#)< value\_type > **vector\_type**

## Public Member Functions

- void [\\_M\\_clear](#) ()
- std::size\_t \* [\\_M\\_get](#) (std::size\_t \_\_sz)
- void [\\_M\\_insert](#) (std::size\_t \* \_\_addr) throw ()

### 4.493.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the `bitmap_allocator`.

Definition at line 516 of file `bitmap_allocator.h`.

### 4.493.2 Member Function Documentation

#### 4.493.2.1 [\\_M\\_clear\(\)](#)

```
void __gnu_cxx::free_list::_M_clear ()
```

This function just clears the internal Free List, and gives back all the memory to the OS.

#### 4.493.2.2 [\\_M\\_get\(\)](#)

```
std::size_t* __gnu_cxx::free_list::_M_get (
 std::size_t __sz)
```

This function gets a block of memory of the specified size from the free list.

#### Parameters

|                   |                                           |
|-------------------|-------------------------------------------|
| <code>__sz</code> | The size in bytes of the memory required. |
|-------------------|-------------------------------------------|

#### Returns

A pointer to the new memory block of size at least equal to that requested.

#### 4.493.2.3 [\\_M\\_insert\(\)](#)

```
void __gnu_cxx::free_list::_M_insert (
 std::size_t * __addr) throw () [inline]
```

This function returns the block of memory to the internal free list.



## Parameters

|                     |                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| <code>__addr</code> | The pointer to the memory block that was given by a call to the <code>_M_get</code> function. |
|---------------------|-----------------------------------------------------------------------------------------------|

Definition at line 626 of file `bitmap_allocator.h`.

The documentation for this class was generated from the following file:

- [bitmap\\_allocator.h](#)

## 4.494 std::from\_chars\_result Struct Reference

## Public Attributes

- `errc` **ec**
- `const char *` **ptr**

## 4.494.1 Detailed Description

Result type of `std::from_chars`.

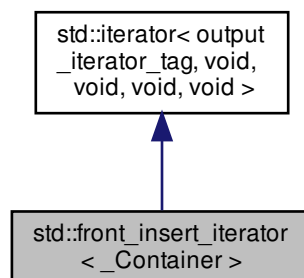
Definition at line 67 of file `charconv`.

The documentation for this struct was generated from the following file:

- [charconv](#)

## 4.495 std::front\_insert\_iterator&lt;\_Container&gt; Class Template Reference

Inheritance diagram for `std::front_insert_iterator<_Container>`:



### Public Types

- typedef `_Container` `container_type`
- typedef void `difference_type`
- typedef `output_iterator_tag` `iterator_category`
- typedef void `pointer`
- typedef void `reference`
- typedef void `value_type`

### Public Member Functions

- constexpr `front_insert_iterator` (`_Container` &\_\_x)
- constexpr `front_insert_iterator` & `operator*` ()
- constexpr `front_insert_iterator` & `operator++` ()
- constexpr `front_insert_iterator` `operator++` (int)
- constexpr `front_insert_iterator` & `operator=` (const typename `_Container::value_type` &\_\_value)
- constexpr `front_insert_iterator` & **`operator=`** (typename `_Container::value_type` &&\_\_value)

### Protected Attributes

- `_Container` \* **`container`**

#### 4.495.1 Detailed Description

```
template<typename _Container>
class std::front_insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using `push_front`.

Tip: Using the `front_inserter` function to create these iterators can save typing.

Definition at line 699 of file `bits/stl_iterator.h`.

#### 4.495.2 Member Typedef Documentation

##### 4.495.2.1 `container_type`

```
template<typename _Container >
typedef _Container std::front_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 707 of file `bits/stl_iterator.h`.

#### 4.495.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file stl\_iterator\_base\_types.h.

#### 4.495.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.495.2.4 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file stl\_iterator\_base\_types.h.

#### 4.495.2.5 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file stl\_iterator\_base\_types.h.

#### 4.495.2.6 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file stl\_iterator\_base\_types.h.

#### 4.495.3 Constructor & Destructor Documentation

##### 4.495.3.1 front\_insert\_iterator()

```
template<typename _Container >
constexpr std::front_insert_iterator< _Container >::front_insert_iterator (
 _Container & __x) [inline], [explicit]
```

The only way to create this iterator is with a container.

Definition at line 716 of file bits/stl\_iterator.h.

#### 4.495.4 Member Function Documentation

##### 4.495.4.1 operator\*()

```
template<typename _Container >
constexpr front_insert_iterator& std::front_insert_iterator< _Container >::operator* () [inline]
```

Simply returns \*this.

Definition at line 758 of file bits/stl\_iterator.h.

##### 4.495.4.2 operator++() [1/2]

```
template<typename _Container >
constexpr front_insert_iterator& std::front_insert_iterator< _Container >::operator++ () [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 764 of file bits/stl\_iterator.h.

##### 4.495.4.3 operator++() [2/2]

```
template<typename _Container >
constexpr front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
 int) [inline]
```

Simply returns \*this. (This iterator does not *move*.)

Definition at line 770 of file bits/stl\_iterator.h.

##### 4.495.4.4 operator=()

```
template<typename _Container >
constexpr front_insert_iterator& std::front_insert_iterator< _Container >::operator= (
 const typename _Container::value_type & __value) [inline]
```

## Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.

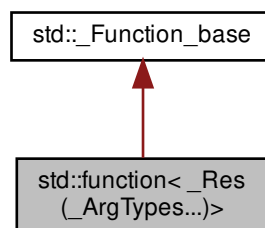
Definition at line 740 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

4.496 `std::function<_Res(_ArgTypes...)>` Class Template Reference

Inheritance diagram for `std::function<_Res(_ArgTypes...)>`:



## Public Types

- `typedef _Res result_type`

### Public Member Functions

- `function` () noexcept
- `function` (nullptr\_t) noexcept
- `function` (const function &\_\_x)
- `function` (function &&\_\_x) noexcept
- `template<typename _Functor , typename = _Requires<__not_<is_same<_Functor, function>>, void>, typename = _Requires<_Callable<_Functor>, void>>`  
`function` (\_Functor)
- `operator bool` () const noexcept
- `_Res operator()` (\_ArgTypes... \_\_args) const
- `function & operator=` (const function &\_\_x)
- `function & operator=` (function &&\_\_x) noexcept
- `function & operator=` (nullptr\_t) noexcept
- `template<typename _Functor >`  
`_Requires< _Callable< typename decay< _Functor >::type >, function & > operator=` (\_Functor &&\_\_f)
- `template<typename _Functor >`  
`function & operator=` (reference\_wrapper< \_Functor > \_\_f) noexcept
- `void swap` (function &\_\_x) noexcept
- `const type_info & target_type` () const noexcept
- `template<typename _Functor >`  
`_Functor * target` () noexcept
- `template<typename _Functor >`  
`const _Functor * target` () const noexcept

### Private Types

- `typedef bool(* _Manager_type)` (\_Any\_data &, const \_Any\_data &, \_Manager\_operation)

### Private Member Functions

- `bool _M_empty` () const

### Private Attributes

- `_Any_data _M_functor`
- `_Manager_type _M_manager`

### Static Private Attributes

- `static const size_t _M_max_align`
- `static const size_t _M_max_size`

#### 4.496.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::function< _Res(_ArgTypes...)>
```

Primary class template for std::function.

Polymorphic function wrapper.

Definition at line 303 of file std\_function.h.

#### 4.496.2 Constructor & Destructor Documentation

##### 4.496.2.1 function() [1/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function () [inline], [noexcept]
```

Default construct creates an empty function call wrapper.

##### Postcondition

!(bool)\*this

Definition at line 330 of file std\_function.h.

##### 4.496.2.2 function() [2/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 nullptr_t) [inline], [noexcept]
```

Creates an empty function call wrapper.

##### Postcondition

!(bool)\*this

Definition at line 337 of file std\_function.h.

##### 4.496.2.3 function() [3/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 const function< _Res(_ArgTypes...)> & __x)
```

Function copy constructor.

**Parameters**

|                   |                                                  |
|-------------------|--------------------------------------------------|
| $\leftrightarrow$ | A function object with identical call signature. |
| <code>__x</code>  |                                                  |

**Postcondition**

```
bool(*this) == bool(__x)
```

The newly-created function contains a copy of the target of `__x` (if it has one).

Definition at line 588 of file `std_function.h`.

**4.496.2.4 function()** [ 4/5]

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
 function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move constructor.

**Parameters**

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| $\leftrightarrow$ | A function object rvalue with identical call signature. |
| <code>__x</code>  |                                                         |

The newly-created function contains the target of `__x` (if it has one).

Definition at line 357 of file `std_function.h`.

**4.496.2.5 function()** [ 5/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor , typename , typename >
std::function< _Res(_ArgTypes...)>::function (
 _Functor __f)
```

Builds a function that targets a copy of the incoming function object.



**Parameters**

|              |                                                                                                                    |
|--------------|--------------------------------------------------------------------------------------------------------------------|
| $\leftarrow$ | A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res. |
| $\_f$        |                                                                                                                    |
| $\leftarrow$ |                                                                                                                    |
| $\_f$        |                                                                                                                    |

The newly-created function object will target a copy of  $\_f$ . If  $\_f$  is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If  $\_f$  is a NULL function pointer or NULL pointer-to-member, the newly-created object will be empty.

If  $\_f$  is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 602 of file `std_function.h`.

References `std::move()`.

**4.496.3 Member Function Documentation****4.496.3.1 operator bool()**

```
template<typename _Res , typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::operator bool () const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

**Returns**

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw an exception.

Definition at line 498 of file `std_function.h`.

**4.496.3.2 operator()()**

```
template<typename _Res , typename... _ArgTypes>
_Res std::function< _Res(_ArgTypes...)>::operator() (
 _ArgTypes... __args) const
```

Invokes the function targeted by `*this`.

**Returns**

the result of the target.

**Exceptions**

|                          |                                |
|--------------------------|--------------------------------|
| <i>bad_function_call</i> | when <code>!(bool)*this</code> |
|--------------------------|--------------------------------|

The function call operator invokes the target function object stored by `this`.

Definition at line 618 of file `std_function.h`.

**4.496.3.3 operator=()** [1/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function< _Res(_ArgTypes...)>::operator= (
 const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function assignment operator.

**Parameters**

|                  |                                           |
|------------------|-------------------------------------------|
| <code>__x</code> | A function with identical call signature. |
|------------------|-------------------------------------------|

**Postcondition**

`(bool)*this == (bool)x`

**Returns**

`*this`

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 396 of file `std_function.h`.

**4.496.3.4 operator=()** [2/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function< _Res(_ArgTypes...)>::operator= (
 function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move-assignment operator.

## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A function rvalue with identical call signature. |
|------------------|--------------------------------------------------|

## Returns

`*this`

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.

If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.

Definition at line 414 of file `std_function.h`.

References `std::move()`.

## 4.496.3.5 operator=() [3/5]

```
template<typename _Res , typename... _ArgTypes>
function& std::function<_Res(_ArgTypes...)>::operator= (
 nullptr_t) [inline], [noexcept]
```

Function assignment to zero.

## Postcondition

`!(bool)*this`

## Returns

`*this`

The target of `*this` is deallocated, leaving it empty.

Definition at line 428 of file `std_function.h`.

## 4.496.3.6 operator=() [4/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
_requires<_Callable<typename decay<_Functor>::type>, function&> std::function<_Res(_ArgTypes...)>::operator= (
 _Functor && __f) [inline]
```

Function assignment to a new target.

**Parameters**

|                 |                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------|
| $\leftarrow$    | A function object that is callable with parameters of type T1, T2, ..., TN and returns a value convertible to Res. |
| $\_ \leftarrow$ |                                                                                                                    |
| $\leftarrow$    |                                                                                                                    |
| $\_ \leftarrow$ |                                                                                                                    |
| $f$             |                                                                                                                    |

**Returns**

`*this`

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a NULL function pointer or NULL pointer-to-member, `this` object will be empty.

If `__f` is a non-NULL function pointer or an object of type `reference_wrapper<F>`, this function will not throw.

Definition at line 457 of file `std_function.h`.

**4.496.3.7 operator=()** [5/5]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
function& std::function< _Res(_ArgTypes...)>::operator= (
 reference_wrapper< _Functor > __f) [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Definition at line 466 of file `std_function.h`.

**4.496.3.8 swap()**

```
template<typename _Res , typename... _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
 function< _Res(_ArgTypes...)> & __x) [inline], [noexcept]
```

Swap the targets of two function objects.

**Parameters**

|                 |                                           |
|-----------------|-------------------------------------------|
| $\_ \leftarrow$ | A function with identical call signature. |
| $\_x$           |                                           |

Swap the targets of `this` function object and `__f`. This function will not throw an exception.

Definition at line 481 of file `std_function.h`.

#### 4.496.3.9 target() [1/2]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
_Functor * std::function< _Res(_ArgTypes...)>::target () [noexcept]
```

Access the stored target function object.

##### Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a NULL pointer.

This function does not throw exceptions.

Definition at line 645 of file `std_function.h`.

#### 4.496.3.10 target() [2/2]

```
template<typename _Res , typename... _ArgTypes>
template<typename _Functor >
const _Functor * std::function< _Res(_ArgTypes...)>::target () const [noexcept]
```

Access the stored target function object.

##### Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a NULL pointer.

This function does not throw exceptions.

Definition at line 656 of file `std_function.h`.

#### 4.496.3.11 `target_type()`

```
template<typename _Res , typename... _ArgTypes>
const type_info & std::function< _Res(_ArgTypes...)>::target_type () const [noexcept]
```

Determine the type of the target of this function object wrapper.

##### Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw an exception.

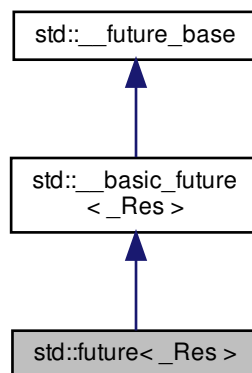
Definition at line 629 of file `std_function.h`.

The documentation for this class was generated from the following file:

- [std\\_function.h](#)

#### 4.497 `std::future<_Res>` Class Template Reference

Inheritance diagram for `std::future<_Res>`:



##### Public Types

- `template<typename _Res>`  
  using `_Ptr` = `unique_ptr<_Res, _Result_base::_Deleter>`
- using `_State_base` = `_State_baseV2`

## Public Member Functions

- `future` (`future` &&\_\_uf) noexcept
- `future` (const `future` &)=delete
- `_Res` `get` ()
- `future` & `operator=` (const `future` &)=delete
- `future` & `operator=` (`future` &&\_\_fut) noexcept
- `shared_future`< `_Res` > `share` () noexcept
- bool `valid` () const noexcept
- void `wait` () const
- template<typename `_Rep` , typename `_Period` >  
`future_status` `wait_for` (const `chrono::duration`< `_Rep`, `_Period` > &\_\_rel) const
- template<typename `_Clock` , typename `_Duration` >  
`future_status` `wait_until` (const `chrono::time_point`< `_Clock`, `_Duration` > &\_\_abs) const

## Static Public Member Functions

- template<typename `_Res` , typename `_Allocator` >  
static `_Ptr`< `_Result_alloc`< `_Res`, `_Allocator` > > `_S_allocate_result` (const `_Allocator` &\_\_a)
- template<typename `_Res` , typename `_Tp` >  
static `_Ptr`< `_Result`< `_Res` > > `_S_allocate_result` (const `std::allocator`< `_Tp` > &\_\_a)
- template<typename `_BoundFn` >  
static `std::shared_ptr`< `_State_base` > `_S_make_async_state` (`_BoundFn` &&\_\_fn)
- template<typename `_BoundFn` >  
static `std::shared_ptr`< `_State_base` > `_S_make_deferred_state` (`_BoundFn` &&\_\_fn)
- template<typename `_Res_ptr` , typename `_BoundFn` >  
static `_Task_setter`< `_Res_ptr`, `_BoundFn` > `_S_task_setter` (`_Res_ptr` &\_\_ptr, `_BoundFn` &\_\_call)

## Protected Types

- typedef `__future_base::_Result`< `_Res` > & `__result_type`

## Protected Member Functions

- `__result_type` `_M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

## Friends

- template<typename `_Fn` , typename... `_Args`>  
`future`< `__async_result_of`< `_Fn`, `_Args`... > > `async` (`launch`, `_Fn` &&, `_Args` &&...)
- template<typename >  
class `packaged_task`
- class `promise`< `_Res` >

#### 4.497.1 Detailed Description

```
template<typename _Res>
class std::future<_Res>
```

Primary template for future.

Definition at line 125 of file future.

#### 4.497.2 Member Typedef Documentation

##### 4.497.2.1 \_Ptr

```
template<typename _Res>
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

#### 4.497.3 Constructor & Destructor Documentation

##### 4.497.3.1 future()

```
template<typename _Res>
std::future<_Res>::future (
 future<_Res> && __uf) [inline], [noexcept]
```

Move constructor.

Definition at line 790 of file future.

#### 4.497.4 Member Function Documentation

##### 4.497.4.1 \_M\_get\_result()

```
template<typename _Res>
__result_type std::__basic_future<_Res>::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file future.



4.497.4.2 `get()`

```
template<typename _Res >
_Res std::future<_Res >::get () [inline]
```

Retrieving the value.

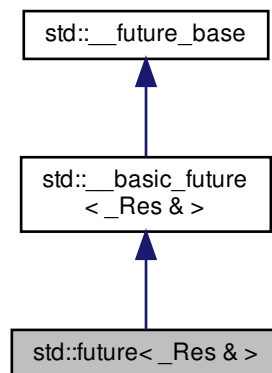
Definition at line 804 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

4.498 `std::future<_Res &>` Class Template Reference

Inheritance diagram for `std::future<_Res &>`:



## Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr<_Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

## Public Member Functions

- `future (future &&__uf) noexcept`
- `future (const future &)=delete`
- `_Res & get ()`
- `future & operator= (const future &)=delete`
- `future & operator= (future &&__fut) noexcept`
- `shared\_future<_Res &> share () noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait_for (const chrono::duration<_Rep, _Period > &__rel) const`
- `future\_status wait_until (const chrono::time\_point<_Clock, _Duration > &__abs) const`

### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr< _Result_alloc< _Res, _Allocator > > _S_allocate_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr< _Result< _Res > > _S_allocate_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_async_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_deferred_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static _Task_setter< _Res_ptr, _BoundFn > _S_task_setter (_Res_ptr &__ptr, _BoundFn &__call)`

### Protected Types

- `typedef __future_base::_Result< _Res & > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename >`  
`class packaged_task`
- `class promise< _Res & >`

#### 4.498.1 Detailed Description

```
template<typename _Res>
class std::future< _Res & >
```

Partial specialization for `future<R&>`

Definition at line 815 of file `future`.

#### 4.498.2 Member Typedef Documentation

#### 4.498.2.1 `_Ptr`

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file `future`.

### 4.498.3 Constructor & Destructor Documentation

#### 4.498.3.1 `future()`

```
template<typename _Res >
std::future<_Res >::future (
 future<_Res > && __uf) [inline], [noexcept]
```

Move constructor.

Definition at line 833 of file `future`.

### 4.498.4 Member Function Documentation

#### 4.498.4.1 `_M_get_result()`

```
__result_type std::__basic_future<_Res >::_M_get_result () const [inline], [protected],
[inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file `future`.

#### 4.498.4.2 `get()`

```
template<typename _Res >
_Res& std::future<_Res >::get () [inline]
```

Retrieving the value.

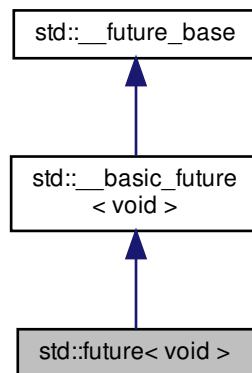
Definition at line 847 of file `future`.

The documentation for this class was generated from the following file:

- `future`

#### 4.499 `std::future< void >` Class Template Reference

Inheritance diagram for `std::future< void >`:



##### Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

##### Public Member Functions

- `future (future &&__uf) noexcept`
- `future (const future &)=delete`
- `void get ()`
- `future & operator= (const future &)=delete`
- `future & operator= (future &&__fut) noexcept`
- `shared\_future< void > share () noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `future\_status wait_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

### Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static _Ptr< _Result_alloc< _Res, _Allocator > > _S_allocate_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static _Ptr< _Result< _Res > > _S_allocate_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_async_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared_ptr< _State_base > _S_make_deferred_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static _Task_setter< _Res_ptr, _BoundFn > _S_task_setter (_Res_ptr &__ptr, _BoundFn &__call)`

### Protected Types

- `typedef __future_base::_Result< void > & __result_type`

### Protected Member Functions

- `__result_type _M_get_result () const`
- `void _M_swap (__basic_future &__that) noexcept`

### Friends

- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > async (launch, _Fn &&, _Args &&...)`
- `template<typename >`  
`class packaged_task`
- `class promise< void >`

#### 4.499.1 Detailed Description

```
template<>
class std::future< void >
```

Explicit specialization for future<void>

Definition at line 858 of file future.

#### 4.499.2 Member Typedef Documentation

#### 4.499.2.1 `_Ptr`

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file `future`.

### 4.499.3 Constructor & Destructor Documentation

#### 4.499.3.1 `future()`

```
std::future< void >::future (
 future< void > && __uf) [inline], [noexcept]
```

Move constructor.

Definition at line 876 of file `future`.

### 4.499.4 Member Function Documentation

#### 4.499.4.1 `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file `future`.

#### 4.499.4.2 `get()`

```
void std::future< void >::get () [inline]
```

Retrieving the value.

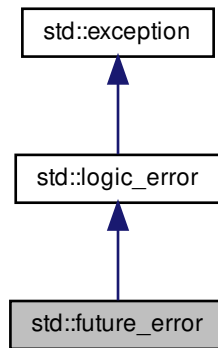
Definition at line 890 of file `future`.

The documentation for this class was generated from the following file:

- `future`

## 4.500 `std::future_error` Class Reference

Inheritance diagram for `std::future_error`:



### Public Member Functions

- **future\_error** ([future\\_errc](#) \_\_errc)
- const [error\\_code](#) & **code** () const noexcept
- virtual const char \* **what** () const noexcept

### Friends

- void **\_\_throw\_future\_error** (int)

#### 4.500.1 Detailed Description

Exception type thrown by futures.

Definition at line 96 of file future.

#### 4.500.2 Member Function Documentation

#### 4.500.2.1 `what()`

```
virtual const char* std::future_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::logic\\_error](#).

The documentation for this class was generated from the following file:

- [future](#)

### 4.501 `std::gamma_distribution<_RealType>` Class Template Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef `_RealType` [result\\_type](#)

#### Public Member Functions

- [gamma\\_distribution](#) ()
- [gamma\\_distribution](#) (`_RealType` \_\_alpha\_val, `_RealType` \_\_beta\_val=`_RealType`(1))
- **[gamma\\_distribution](#)** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- `_RealType` [alpha](#) () const
- `_RealType` [beta](#) () const
- [result\\_type](#) [max](#) () const
- [result\\_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()



## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::gamma_distribution< _RealType1 > &__x)`
- `bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::gamma_distribution< _RealType1 > &__x)`

## 4.501.1 Detailed Description

```
template<typename _RealType = double>
class std::gamma_distribution< _RealType >
```

A gamma continuous distribution for random numbers.

The formula for the gamma probability density function is:

$$p(x|\alpha, \beta) = \frac{1}{\beta\Gamma(\alpha)} (x/\beta)^{\alpha-1} e^{-x/\beta}$$

Definition at line 2402 of file random.h.

## 4.501.2 Member Typedef Documentation

## 4.501.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 2405 of file random.h.

## 4.501.3 Constructor &amp; Destructor Documentation

## 4.501.3.1 gamma\_distribution() [1/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution () [inline]
```

Constructs a gamma distribution with parameters 1 and 1.

Definition at line 2458 of file random.h.

#### 4.501.3.2 `gamma_distribution()` [2/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
 _RealType __alpha_val,
 _RealType __beta_val = _RealType(1)) [inline], [explicit]
```

Constructs a gamma distribution with parameters  $\alpha$  and  $\beta$ .

Definition at line 2465 of file random.h.

#### 4.501.4 Member Function Documentation

##### 4.501.4.1 `alpha()`

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::alpha () const [inline]
```

Returns the  $\alpha$  of the distribution.

Definition at line 2486 of file random.h.

##### 4.501.4.2 `beta()`

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::beta () const [inline]
```

Returns the  $\beta$  of the distribution.

Definition at line 2493 of file random.h.

##### 4.501.4.3 `max()`

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2522 of file random.h.

## 4.501.4.4 min()

```
template<typename _RealType = double>
result_type std::gamma_distribution<_RealType>::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2515 of file random.h.

## 4.501.4.5 operator() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::gamma_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 2530 of file random.h.

Referenced by std::gamma\_distribution<result\_type>::operator()().

## 4.501.4.6 operator() [2/2]

```
template<typename _RealType >
template<typename _UniformRandomNumberGenerator >
gamma_distribution<_RealType>::result_type std::gamma_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.

Definition at line 2321 of file bits/random.tcc.

## 4.501.4.7 param() [1/2]

```
template<typename _RealType = double>
param_type std::gamma_distribution<_RealType>::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2500 of file random.h.

Referenced by std::chi\_squared\_distribution<\_RealType>::param().

## 4.501.4.8 param() [2/2]

```
template<typename _RealType = double>
void std::gamma_distribution<_RealType>::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2508 of file random.h.

#### 4.501.4.9 `reset()`

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 2479 of file random.h.

Referenced by `std::chi_squared_distribution< _RealType >::reset()`, `std::fisher_f_distribution< _RealType >::reset()`, `std::student_t_distribution< _RealType >::reset()`, and `std::negative_binomial_distribution< _IntType >::reset()`.

#### 4.501.5 Friends And Related Function Documentation

##### 4.501.5.1 `operator<<`

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::gamma_distribution< _RealType1 > & __x) [friend]
```

Inserts a `gamma_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <code>__os</code> | An output stream.                                             |
| <code>__x</code>  | A <code>gamma_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

## 4.501.5.2 operator==

```
template<typename _RealType = double>
bool operator== (
 const gamma_distribution< _RealType > & __d1,
 const gamma_distribution< _RealType > & __d2) [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2566 of file random.h.

## 4.501.5.3 operator&gt;&gt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::gamma_distribution< _RealType1 > & __x) [friend]
```

Extracts a [gamma\\_distribution](#) random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                      |                                                                      |
|----------------------|----------------------------------------------------------------------|
| <a href="#">__is</a> | An input stream.                                                     |
| <a href="#">__x</a>  | A <a href="#">gamma_distribution</a> random number generator engine. |

## Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.502 std::geometric\_distribution&lt; \_IntType &gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- **geometric\_distribution** (double \_\_p)
- **geometric\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [geometric\\_distribution](#) &\_\_d1, const [geometric\\_distribution](#) &\_\_d2)

### 4.502.1 Detailed Description

```
template<typename _IntType = int>
class std::geometric_distribution< _IntType >
```

A discrete geometric random number distribution.

The formula for the geometric probability density function is  $p(i|p) = p(1 - p)^i$  where  $p$  is the parameter of the distribution.

Definition at line 3978 of file random.h.

### 4.502.2 Member Typedef Documentation

#### 4.502.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::geometric_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 3981 of file random.h.

#### 4.502.3 Member Function Documentation

##### 4.502.3.1 max()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4080 of file random.h.

References std::numeric\_limits< \_Tp >::max().

##### 4.502.3.2 min()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4073 of file random.h.

##### 4.502.3.3 operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::geometric_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 4088 of file random.h.

#### 4.502.3.4 p()

```
template<typename _IntType = int>
double std::geometric_distribution< _IntType >::p () const [inline]
```

Returns the distribution parameter p.

Definition at line 4051 of file random.h.

#### 4.502.3.5 param() [1/2]

```
template<typename _IntType = int>
param_type std::geometric_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4058 of file random.h.

Referenced by std::operator>>().

#### 4.502.3.6 param() [2/2]

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4066 of file random.h.

#### 4.502.3.7 reset()

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the geometric distribution.

Definition at line 4045 of file random.h.



#### 4.502.4 Friends And Related Function Documentation

##### 4.502.4.1 `operator==`

```
template<typename _IntType = int>
bool operator== (
 const geometric_distribution< _IntType > & __d1,
 const geometric_distribution< _IntType > & __d2) [friend]
```

Return true if two geometric distributions have the same parameters.

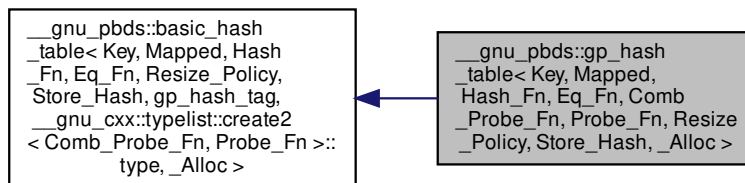
Definition at line 4123 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.503 `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >`:



#### Public Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn**
- typedef [gp\\_hash\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Hash\_Fn **hash\_fn**
- typedef Probe\_Fn **probe\_fn**
- typedef Resize\_Policy **resize\_policy**

## Public Member Functions

- [gp\\_hash\\_table](#) ()
- [gp\\_hash\\_table](#) (const hash\_fn &h)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- [gp\\_hash\\_table](#) (const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p)
- template<typename It >  
[gp\\_hash\\_table](#) (It first, It last, const hash\_fn &h, const eq\_fn &e, const comb\_probe\_fn &cp, const probe\_fn &p, const resize\_policy &rp)
- **gp\_hash\_table** (const [gp\\_hash\\_table](#) &other)
- [gp\\_hash\\_table](#) & **operator=** (const [gp\\_hash\\_table](#) &other)
- void **swap** ([gp\\_hash\\_table](#) &other)

## 4.503.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn
= typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn
= typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >
```

A general-probing hash-based associative container.

## Template Parameters

|                      |                                                                                                                                                                                                  |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                        |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                        |
| <i>Hash_Fn</i>       | Hashing functor.                                                                                                                                                                                 |
| <i>Eq_Fn</i>         | Equal functor.                                                                                                                                                                                   |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. |
| <i>Probe_Fn</i>      | Probe functor.                                                                                                                                                                                   |
| <i>Resize_Policy</i> | Resizes hash.                                                                                                                                                                                    |
| <i>Store_Hash</i>    | Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true                                          |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                  |

Base tag choices are: `gp_hash_tag`.

Base is `basic_hash_table`.

Definition at line 368 of file `assoc_container.hpp`.

#### 4.503.2 Constructor & Destructor Documentation

##### 4.503.2.1 `gp_hash_table()` [1/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table () [inline]
```

Default constructor.

Definition at line 382 of file `assoc_container.hpp`.

##### 4.503.2.2 `gp_hash_table()` [2/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 386 of file `assoc_container.hpp`.

4.503.2.3 `gp_hash_table()` [3/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 393 of file `assoc_container.hpp`.

4.503.2.4 `gp_hash_table()` [4/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 401 of file `assoc_container.hpp`.

4.503.2.5 `gp_hash_table()` [5/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 410 of file `assoc_container.hpp`.

4.503.2.6 `gp_hash_table()` [6/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p,
 const resize_policy & rp) [inline]
```

Constructor taking some policy objects. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `Resize_Policy` object of the container object.

Definition at line 422 of file `assoc_container.hpp`.

4.503.2.7 `gp_hash_table()` [7/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 430 of file `assoc_container.hpp`.

4.503.2.8 `gp_hash_table()` [8/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
```

```
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object.

Definition at line 438 of file `assoc_container.hpp`.

#### 4.503.2.9 gp\_hash\_table() [9/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, and `r_eq_fn` will be copied by the `eq_fn` object of the container object.

Definition at line 449 of file `assoc_container.hpp`.

#### 4.503.2.10 gp\_hash\_table() [10/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, and `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object.

Definition at line 461 of file `assoc_container.hpp`.

4.503.2.11 `gp_hash_table()` [11/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_probe_fn` object of the container object, and `r_probe_fn` will be copied by the `probe_fn` object of the container object.

Definition at line 475 of file `assoc_container.hpp`.

4.503.2.12 `gp_hash_table()` [12/12]

```
template<typename Key , typename Mapped , typename Hash_Fn = typename detail::default_hash_fn<↵
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↵
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↵
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
 It first,
 It last,
 const hash_fn & h,
 const eq_fn & e,
 const comb_probe_fn & cp,
 const probe_fn & p,
 const resize_policy & rp) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object. `r_hash_fn` will be copied by the `hash_fn` object of the container object, `r_eq_fn` will be copied by the `eq_fn` object of the container object, `r_comb_probe_fn` will be copied by the `comb_↵` `probe_fn` object of the container object, `r_probe_fn` will be copied by the `probe_fn` object of the container object, and `r_resize_policy` will be copied by the `resize_policy` object of the container object.

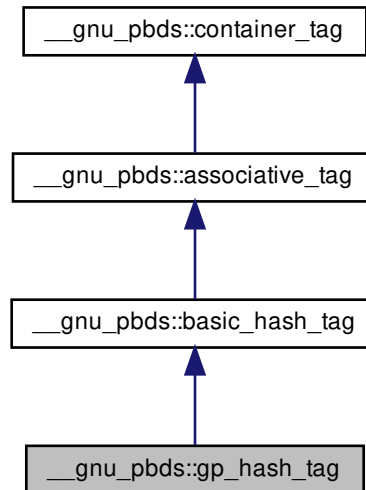
Definition at line 491 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

#### 4.504 \_\_gnu\_pbds::gp\_hash\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::gp\_hash\_tag:



##### 4.504.1 Detailed Description

General-probing hash.

Definition at line 144 of file `tag_and_trait.hpp`.

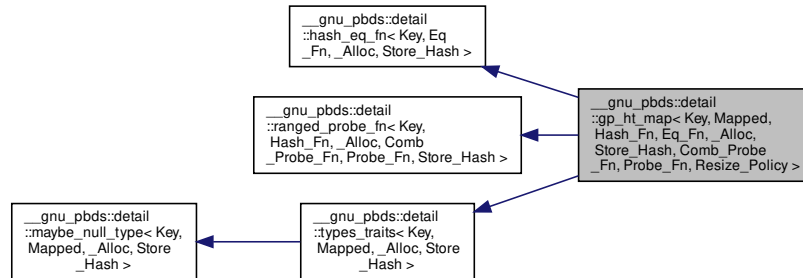
The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)



4.505 `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`:



## Public Types

- enum { **store\_hash** }
- typedef `_Alloc` **allocator\_type**
- typedef `Comb_Probe_Fn` **comb\_probe\_fn**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `Hash_Fn` **hash\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `Probe_Fn` **probe\_fn**
- typedef `traits_base::reference` **reference**
- typedef `Resize_Policy` **resize\_policy**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **gp\_ht\_map** (const [gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)
- **gp\_ht\_map** (const Hash\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &)
- **gp\_ht\_map** (const Hash\_Fn &, const Eq\_Fn &, const Comb\_Probe\_Fn &, const Probe\_Fn &, const Resize\_Policy &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It >  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- point\_iterator **find\_end** ()
- point\_const\_iterator **find\_end** () const
- Comb\_Probe\_Fn & [get\\_comb\\_probe\\_fn](#) ()
- const Comb\_Probe\_Fn & [get\\_comb\\_probe\\_fn](#) () const
- Eq\_Fn & [get\\_eq\\_fn](#) ()
- const Eq\_Fn & [get\\_eq\\_fn](#) () const
- Hash\_Fn & [get\\_hash\\_fn](#) ()
- const Hash\_Fn & [get\\_hash\\_fn](#) () const
- Probe\_Fn & [get\\_probe\\_fn](#) ()
- const Probe\_Fn & [get\\_probe\\_fn](#) () const
- Resize\_Policy & [get\\_resize\\_policy](#) ()
- const Resize\_Policy & [get\\_resize\\_policy](#) () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_val)
- size\_type **max\_size** () const
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **swap** ([gp\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy > &)

## Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

## Friends

- |   |       |                        |
|---|-------|------------------------|
| • | class | <b>const_iterator_</b> |
| • | class | <b>iterator_</b>       |

#### 4.505.1 Detailed Description

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename
Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy
>
```

A general-probing hash-based container.

##### Template Parameters

|                      |                                                                                                                                                                                                                                                          |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                                                                                                                                                                                                |
| <i>Mapped</i>        | Map type.                                                                                                                                                                                                                                                |
| <i>Hash_Fn</i>       | Hashing functor. Default is <code>__gnu_cxx::hash</code> .                                                                                                                                                                                               |
| <i>Eq_Fn</i>         | Equal functor. Default <code>std::equal_to&lt;Key&gt;</code>                                                                                                                                                                                             |
| <i>_Alloc</i>        | Allocator type.                                                                                                                                                                                                                                          |
| <i>Store_Hash</i>    | If key type stores extra metadata. Defaults to false.                                                                                                                                                                                                    |
| <i>Comb_Probe_Fn</i> | Combining probe functor. If <i>Hash_Fn</i> is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default <code>direct_mask_range_hashing</code> . |
| <i>Probe_Fn</i>      | Probe functor. Defaults to <code>linear_probe_fn</code> , also <code>quadratic_probe_fn</code> .                                                                                                                                                         |
| <i>Resize_Policy</i> | Resizes hash. Defaults to <code>hash_standard_resize_policy</code> , using <code>hash_exponential_size_policy</code> and <code>hash_load_check_resize_trigger</code> .                                                                                   |

Bases are: `detail::hash_eq_fn`, `Resize_Policy`, `detail::ranged_probe_fn`, `detail::types_traits`. (Optional: `detail::debug_↵map_base`.)

Definition at line 142 of file `gp_ht_map.hpp`.

#### 4.505.2 Member Enumeration Documentation

##### 4.505.2.1 anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

Definition at line 209 of file `gp_ht_map.hpp`.

#### 4.505.3 Member Function Documentation

#### 4.505.3.1 empty()

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::empty () const [inline]
```

True if size() == 0.

Definition at line 61 of file gp\_ht\_map.hpp.

#### 4.505.3.2 get\_comb\_probe\_fn() [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()
```

Return current comb\_probe\_fn.

Definition at line 84 of file gp\_ht\_map.hpp.

#### 4.505.3.3 get\_comb\_probe\_fn() [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const
```

Return current const comb\_probe\_fn.

Definition at line 90 of file gp\_ht\_map.hpp.

#### 4.505.3.4 get\_eq\_fn() [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()
```

Return current eq\_fn.

Definition at line 60 of file gp\_ht\_map.hpp.

#### 4.505.3.5 `get_eq_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq\_fn.

Definition at line 66 of file `gp_ht_map.hpp`.

#### 4.505.3.6 `get_hash_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash\_fn.

Definition at line 48 of file `gp_ht_map.hpp`.

#### 4.505.3.7 `get_hash_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash\_fn.

Definition at line 54 of file `gp_ht_map.hpp`.

#### 4.505.3.8 `get_probe_fn()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↵
Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()
```

Return current probe\_fn.

Definition at line 72 of file `gp_ht_map.hpp`.

**4.505.3.9** `get_probe_fn()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const
```

Return current const probe\_fn.

Definition at line 78 of file `gp_ht_map.hpp`.

**4.505.3.10** `get_resize_policy()` [1/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize\_policy.

Definition at line 96 of file `gp_ht_map.hpp`.

**4.505.3.11** `get_resize_policy()` [2/2]

```
template<typename Key , typename Mapped , typename Hash_Fn , typename Eq_Fn , typename _Alloc ,
bool Store_Hash, typename Comb_Probe_Fn , typename Probe_Fn , typename Resize_Policy >
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↵
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const
```

Return current const resize\_policy.

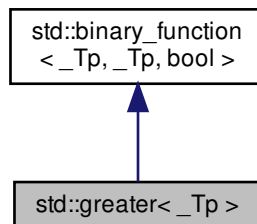
Definition at line 102 of file `gp_ht_map.hpp`.

The documentation for this class was generated from the following file:

- [gp\\_ht\\_map.hpp](#)

**4.506** `std::greater<_Tp>` Struct Template Reference

Inheritance diagram for `std::greater<_Tp>`:



### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- constexpr bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

#### 4.506.1 Detailed Description

```
template<typename _Tp>
struct std::greater< _Tp >
```

One of the [comparison functors](#).

Definition at line 337 of file stl\_function.h.

#### 4.506.2 Member Typedef Documentation

##### 4.506.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 4.506.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

[result\\_type](#) is the return type

Definition at line 127 of file stl\_function.h.

#### 4.506.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.507 std::greater< void > Struct Template Reference

#### Public Types

- typedef \_\_is\_transparent **is\_transparent**

#### Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) > std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) > std::forward< \_Up >(\_\_u))
- template<typename \_Tp, typename \_Up >  
constexpr bool **operator()** (\_Tp \*\_\_t, \_Up \*\_\_u) const noexcept

#### 4.507.1 Detailed Description

```
template<>
struct std::greater< void >
```

One of the [comparison functors](#).

Definition at line 516 of file stl\_function.h.

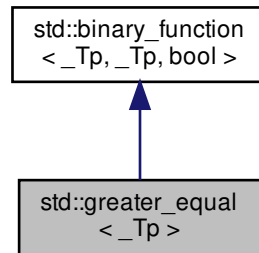
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



## 4.508 std::greater\_equal<\_Tp> Struct Template Reference

Inheritance diagram for std::greater\_equal<\_Tp>:



### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

### Public Member Functions

- constexpr bool **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

#### 4.508.1 Detailed Description

```
template<typename _Tp>
struct std::greater_equal<_Tp>
```

One of the [comparison functors](#).

Definition at line 343 of file stl\_function.h.

#### 4.508.2 Member Typedef Documentation

#### 4.508.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.508.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.508.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.509 std::greater\_equal< void > Struct Template Reference

#### Public Types

- typedef \_\_is\_transparent **is\_transparent**

#### Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) >=std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) >=std::forward< \_Up >(\_\_u))
- template<typename \_Tp, typename \_Up >  
constexpr bool **operator()** (\_Tp \*\_\_t, \_Up \*\_\_u) const noexcept

## 4.509.1 Detailed Description

```
template<>
struct std::greater_equal< void >
```

One of the [comparison functors](#).

Definition at line 640 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.510 `__gnu_cxx::random_condition::group_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

## Public Member Functions

- **`group_adjustor`** (`size_t` size)

## 4.510.1 Detailed Description

Group condition.

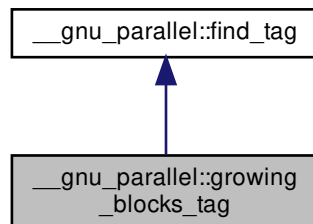
Definition at line 518 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.511 `__gnu_parallel::growing_blocks_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::growing_blocks_tag`:



#### 4.511.1 Detailed Description

Selects the growing block size variant for `std::find()`.

See also

`_GLIBCXX_FIND_GROWING_BLOCKS`

Definition at line 174 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.512 std::gslice Class Reference

##### Public Member Functions

- [gslice](#) ()
- [gslice](#) (size\_t \_\_o, const [valarray](#)< size\_t > &\_\_l, const [valarray](#)< size\_t > &\_\_s)
- [gslice](#) (const [gslice](#) &)
- [~gslice](#) ()
- [gslice](#) & [operator=](#) (const [gslice](#) &)
- [valarray](#)< size\_t > [size](#) () const
- size\_t [start](#) () const
- [valarray](#)< size\_t > [stride](#) () const

##### Friends

- `template<typename _Tp >`  
class **[valarray](#)**

#### 4.512.1 Detailed Description

Class defining multi-dimensional subset of an array.

The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size.

For example, if you have `offset==3`, `stride[0]==11`, `size[1]==3`, `stride[1]==3`, then `slice[0,0]==array[3]`, `slice[0,1]==array[6]`, `slice[0,2]==array[9]`, `slice[1,0]==array[14]`, `slice[1,1]==array[17]`, `slice[1,2]==array[20]`.

Definition at line 64 of file `gslice.h`.

The documentation for this class was generated from the following file:

- [gslice.h](#)

## 4.513 std::gslice\_array&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- `gslice_array` (const `gslice_array` &)
- void `operator%=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator%=(const _Expr< _Dom, _Tp > &)` const
- void `operator&=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator&=(const _Expr< _Dom, _Tp > &)` const
- void `operator*=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator*=(const _Expr< _Dom, _Tp > &)` const
- void `operator+=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator+=(const _Expr< _Dom, _Tp > &)` const
- void `operator-=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator-=(const _Expr< _Dom, _Tp > &)` const
- void `operator/=(const valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator/=(const _Expr< _Dom, _Tp > &)` const
- void `operator<=<=` (const `valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator<=<=` (const `_Expr< _Dom, _Tp > &)` const
- `gslice_array` & `operator=` (const `gslice_array` &)
- void `operator=` (const `valarray< _Tp > &)` const
- void `operator=` (const `_Tp` &) const
- template<class `_Dom` >  
void `operator=` (const `_Expr< _Dom, _Tp > &)` const
- void `operator>>=` (const `valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator>>=` (const `_Expr< _Dom, _Tp > &)` const
- void `operator^=` (const `valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator^=` (const `_Expr< _Dom, _Tp > &)` const
- void `operator|=` (const `valarray< _Tp > &)` const
- template<class `_Dom` >  
void `operator|=` (const `_Expr< _Dom, _Tp > &)` const

## Friends

- class `valarray< _Tp >`

#### 4.513.1 Detailed Description

```
template<typename _Tp>
class std::gslice_array< _Tp >
```

Reference to multi-dimensional subset of an array.

A `gslice_array` is a reference to the actual elements of an array specified by a `gslice`. The way to get a `gslice_array` is to call `operator[]`(`gslice`) on a `valarray`. The returned `gslice_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`. For example, `operator+=(valarray)` will add values to the subset of elements in the underlying `valarray` this `gslice_array` refers to.

##### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

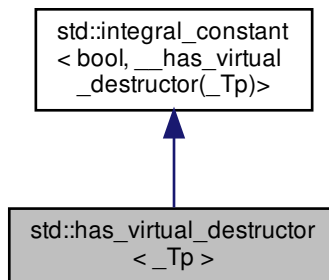
Definition at line 93 of file `valarray`.

The documentation for this class was generated from the following files:

- [valarray](#)
- [gslice\\_array.h](#)

#### 4.514 std::has\_virtual\_destructor< \_Tp > Struct Template Reference

Inheritance diagram for `std::has_virtual_destructor< _Tp >`:



##### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.514.1 Detailed Description

```
template<typename _Tp>
struct std::has_virtual_destructor<_Tp>
```

has\_virtual\_destructor

Definition at line 1338 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.515 `std::hash<_Tp>` Struct Template Reference

Inherits `std::__hash_enum<_Tp, bool>`.

## 4.515.1 Detailed Description

```
template<typename _Tp>
struct std::hash<_Tp>
```

Primary class template hash.

Primary class template hash, usable for enum types only.

Definition at line 110 of file `typeindex`.

The documentation for this struct was generated from the following file:

- [typeindex](#)

4.516 `std::hash<__debug::bitset<_Nb>>` Struct Template Reference

Inherits `std::__hash_base<_Result, _Arg>`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (const [\\_\\_debug::bitset](#)< `_Nb` > &`__b`) const noexcept

##### 4.516.1 Detailed Description

```
template<size_t _Nb>
struct std::hash<__debug::bitset< _Nb > >
```

`std::hash` specialization for `bitset`.

Definition at line 418 of file `debug/bitset`.

The documentation for this struct was generated from the following file:

- [debug/bitset](#)

##### 4.517 `std::hash<__debug::vector< bool, _Alloc > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (const [\\_\\_debug::vector](#)< `bool`, `_Alloc` > &`__b`) const noexcept

##### 4.517.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<__debug::vector< bool, _Alloc > >
```

`std::hash` specialization for `vector<bool>`.

Definition at line 795 of file `debug/vector`.

The documentation for this struct was generated from the following file:

- [debug/vector](#)



## 4.518 `std::hash< __gnu_cxx::__u16vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const \_\_gnu\_cxx::\_\_u16vstring &__s) const` noexcept

#### 4.518.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__u16vstring >
```

`std::hash` specialization for `__u16vstring`.

Definition at line 2939 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

## 4.519 `std::hash< __gnu_cxx::__u32vstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const \_\_gnu\_cxx::\_\_u32vstring &__s) const` noexcept

#### 4.519.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__u32vstring >
```

std::hash specialization for \_\_u32vstring.

Definition at line 2950 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 4.520 std::hash< \_\_gnu\_cxx::\_\_vstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

##### Public Member Functions

- `size_t operator() (const __gnu_cxx::__vstring &__s) const` noexcept

#### 4.520.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__vstring >
```

std::hash specialization for \_\_vstring.

Definition at line 2916 of file vstring.h.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

#### 4.521 std::hash< \_\_gnu\_cxx::\_\_wvstring > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const `__gnu_cxx::__wvstring` &\_\_s) const noexcept

## 4.521.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::__wvstring >
```

`std::hash` specialization for `__wvstring`.

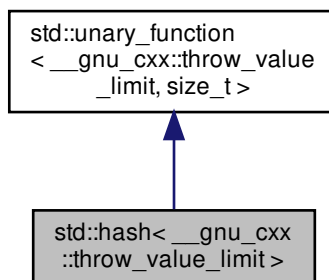
Definition at line 2927 of file `vstring.h`.

The documentation for this struct was generated from the following file:

- [vstring.h](#)

4.522 `std::hash< __gnu_cxx::throw_value_limit >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_limit >`:



## Public Types

- typedef `__gnu_cxx::throw_value_limit` **argument\_type**
- typedef `size_t` **result\_type**

## Public Member Functions

- `size_t operator()` (const [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#) &\_\_val) const

### 4.522.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::throw_value_limit >
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_limit`.

Definition at line 977 of file `throw_allocator.h`.

### 4.522.2 Member Typedef Documentation

#### 4.522.2.1 `argument_type`

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit , size_t
>::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.522.2.2 `result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit , size_t >::result_type [inherited]
```

`result_type` is the return type

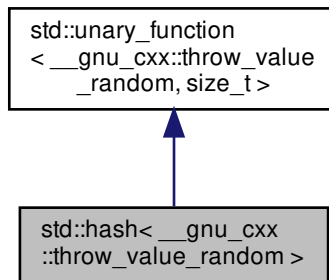
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.523 `std::hash< __gnu_cxx::throw_value_random >` Struct Template Reference

Inheritance diagram for `std::hash< __gnu_cxx::throw_value_random >`:



## Public Types

- typedef `__gnu_cxx::throw_value_random` `argument_type`
- typedef `size_t` `result_type`

## Public Member Functions

- `size_t operator()` (const `__gnu_cxx::throw_value_random` &`__val`) const

## 4.523.1 Detailed Description

```
template<>
struct std::hash< __gnu_cxx::throw_value_random >
```

Explicit specialization of `std::hash` for `__gnu_cxx::throw_value_random`.

Definition at line 993 of file `throw_allocator.h`.

## 4.523.2 Member Typedef Documentation

#### 4.523.2.1 `argument_type`

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random , size_t
>::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.523.2.2 `result_type`

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random , size_t >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

### 4.524 `std::hash< __shared_ptr< _Tp, _Lp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **`argument_type`**
- typedef `_Result` **`result_type`**

#### Public Member Functions

- `size_t operator()` (const `__shared_ptr< _Tp, _Lp >` &`__s`) const noexcept

#### 4.524.1 Detailed Description

```
template<typename _Tp, _Lock_policy _Lp>
struct std::hash< __shared_ptr< _Tp, _Lp > >
```

`std::hash` specialization for `__shared_ptr`.

Definition at line 1884 of file `shared_ptr_base.h`.

The documentation for this struct was generated from the following file:

- [shared\\_ptr\\_base.h](#)

## 4.525 std::hash< \_Tp \* > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

### Public Member Functions

- size\_t **operator()** (\_Tp \*\_\_p) const noexcept

#### 4.525.1 Detailed Description

```
template<typename _Tp>
struct std::hash< _Tp * >
```

Partial specializations for pointer types.

Definition at line 106 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.526 std::hash< bool > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

### Public Member Functions

- size\_t **operator()** (bool \_\_val) const noexcept

#### 4.526.1 Detailed Description

```
template<>
struct std::hash< bool >
```

Explicit specialization for bool.

Definition at line 124 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.527 std::hash< char > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

##### Public Member Functions

- size\_t **operator() (char \_\_val)** const noexcept

#### 4.527.1 Detailed Description

```
template<>
struct std::hash< char >
```

Explicit specialization for char.

Definition at line 127 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.528 std::hash< char16\_t > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.



#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator() (char16_t __val) const` noexcept

##### 4.528.1 Detailed Description

```
template<>
struct std::hash< char16_t >
```

Explicit specialization for `char16_t`.

Definition at line 144 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.529 `std::hash< char32_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator() (char32_t __val) const` noexcept

##### 4.529.1 Detailed Description

```
template<>
struct std::hash< char32_t >
```

Explicit specialization for `char32_t`.

Definition at line 147 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.530 `std::hash< double >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`double __val`) `const noexcept`

#### 4.530.1 Detailed Description

```
template<>
struct std::hash< double >
```

Specialization for `double`.

Definition at line 243 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

### 4.531 `std::hash< error_code >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`const error\_code &__e`) `const noexcept`

## 4.531.1 Detailed Description

```
template<>
struct std::hash< error_code >
```

`std::hash` specialization for `error_code`.

Definition at line 479 of file `system_error`.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

4.532 `std::hash< experimental::optional< _Tp > >` Struct Template Reference

## Public Types

- using **argument\_type** = `experimental::optional< _Tp >`
- using **result\_type** = `size_t`

## Public Member Functions

- `size_t` **operator()** (const `experimental::optional< _Tp > &__t`) const noexcept(noexcept([hash< \\_Tp > {}](#)(\*\_\_t)))

## 4.532.1 Detailed Description

```
template<typename _Tp>
struct std::hash< experimental::optional< _Tp > >
```

`std::hash` partial specialization for `experimental::optional`

Definition at line 922 of file `experimental/optional`.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

4.533 `std::hash< experimental::shared_ptr< _Tp > >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator()` (const experimental::shared\_ptr< \_Tp > &\_\_s) const noexcept

#### 4.533.1 Detailed Description

```
template<typename _Tp>
struct std::hash< experimental::shared_ptr< _Tp > >
```

std::hash specialization for shared\_ptr.

Definition at line 671 of file experimental/bits/shared\_ptr.h.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

#### 4.534 std::hash< float > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

### Public Member Functions

- `size_t operator()` (float \_\_val) const noexcept

#### 4.534.1 Detailed Description

```
template<>
struct std::hash< float >
```

Specialization for float.

Definition at line 231 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.535 `std::hash< int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (int __val) const` noexcept

#### 4.535.1 Detailed Description

```
template<>
struct std::hash< int >
```

Explicit specialization for `int`.

Definition at line 153 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.536 `std::hash< long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (long __val) const` noexcept

#### 4.536.1 Detailed Description

```
template<>
struct std::hash< long >
```

Explicit specialization for long.

Definition at line 156 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.537 std::hash< long double > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

##### Public Member Functions

- size\_t **operator() (long double \_\_val)** const noexcept

#### 4.537.1 Detailed Description

```
template<>
struct std::hash< long double >
```

Specialization for long double.

Definition at line 255 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.538 std::hash< long long > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- size\_t **operator()** (long long \_\_val) const noexcept

##### 4.538.1 Detailed Description

```
template<>
struct std::hash< long long >
```

Explicit specialization for long long.

Definition at line 159 of file functional\_hash.h.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

## 4.539 std::hash< shared\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

#### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- size\_t **operator()** (const [shared\\_ptr](#)< \_Tp > &\_\_s) const noexcept

##### 4.539.1 Detailed Description

```
template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >
```

std::hash specialization for shared\_ptr.

Definition at line 881 of file bits/shared\_ptr.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

#### 4.540 `std::hash< short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator() (short __val) const` noexcept

##### 4.540.1 Detailed Description

```
template<>
struct std::hash< short >
```

Explicit specialization for short.

Definition at line 150 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.541 `std::hash< signed char >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator() (signed char __val) const` noexcept



## 4.541.1 Detailed Description

```
template<>
struct std::hash< signed char >
```

Explicit specialization for signed char.

Definition at line 130 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.542 `std::hash< string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t operator()` (const [string](#) &\_\_s) const noexcept

## 4.542.1 Detailed Description

```
template<>
struct std::hash< string >
```

`std::hash` specialization for string.

Definition at line 6808 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

4.543 `std::hash< thread::id >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (const [thread::id](#) &\_\_id) const noexcept

##### 4.543.1 Detailed Description

```
template<>
struct std::hash< thread::id >
```

std::hash specialization for thread::id.

Definition at line 338 of file thread.

The documentation for this struct was generated from the following file:

- [thread](#)

#### 4.544 std::hash< type\_index > Struct Template Reference

##### Public Types

- typedef [type\\_index](#) **argument\_type**
- typedef `size_t` **result\_type**

##### Public Member Functions

- `size_t operator()` (const [type\\_index](#) &\_\_ti) const noexcept

##### 4.544.1 Detailed Description

```
template<>
struct std::hash< type_index >
```

std::hash specialization for type\_index.

Definition at line 114 of file typeindex.

The documentation for this struct was generated from the following file:

- [typeindex](#)

## 4.545 `std::hash< u16string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const u16string &__s) const` noexcept

#### 4.545.1 Detailed Description

```
template<>
struct std::hash< u16string >
```

`std::hash` specialization for `u16string`.

Definition at line 6857 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

## 4.546 `std::hash< u32string >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

### Public Member Functions

- `size_t operator() (const u32string &__s) const` noexcept

#### 4.546.1 Detailed Description

```
template<>
struct std::hash< u32string >
```

std::hash specialization for u32string.

Definition at line 6872 of file basic\_string.h.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)

#### 4.547 std::hash< unique\_ptr< \_Tp, \_Dp > > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >, and \_\_uniq\_ptr\_hash< unique\_ptr< \_Tp, \_Dp > >.

##### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### 4.547.1 Detailed Description

```
template<typename _Tp, typename _Dp>
struct std::hash< unique_ptr< _Tp, _Dp > >
```

std::hash specialization for unique\_ptr.

Definition at line 933 of file unique\_ptr.h.

The documentation for this struct was generated from the following file:

- [unique\\_ptr.h](#)

#### 4.548 std::hash< unsigned char > Struct Template Reference

Inherits std::\_\_hash\_base< \_Result, \_Arg >.

##### Public Types

- typedef \_Arg **argument\_type**
- typedef \_Result **result\_type**

#### Public Member Functions

- `size_t operator()` (unsigned char \_\_val) const noexcept

##### 4.548.1 Detailed Description

`template<>`  
`struct std::hash< unsigned char >`

Explicit specialization for unsigned char.

Definition at line 133 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.549 `std::hash< unsigned int >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- `typedef _Arg argument_type`
- `typedef _Result result_type`

#### Public Member Functions

- `size_t operator()` (unsigned int \_\_val) const noexcept

##### 4.549.1 Detailed Description

`template<>`  
`struct std::hash< unsigned int >`

Explicit specialization for unsigned int.

Definition at line 165 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.550 `std::hash< unsigned long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator() (unsigned long __val) const` noexcept

##### 4.550.1 Detailed Description

```
template<>
struct std::hash< unsigned long >
```

Explicit specialization for unsigned long.

Definition at line 168 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.551 `std::hash< unsigned long long >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator() (unsigned long long __val) const` noexcept

## 4.551.1 Detailed Description

```
template<>
struct std::hash< unsigned long long >
```

Explicit specialization for unsigned long long.

Definition at line 171 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.552 `std::hash< unsigned short >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

## Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

## Public Member Functions

- `size_t` **operator()** (`unsigned short __val`) `const noexcept`

## 4.552.1 Detailed Description

```
template<>
struct std::hash< unsigned short >
```

Explicit specialization for unsigned short.

Definition at line 162 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

4.553 `std::hash< wchar_t >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`wchar_t __val`) `const noexcept`

##### 4.553.1 Detailed Description

```
template<>
struct std::hash< wchar_t >
```

Explicit specialization for `wchar_t`.

Definition at line 136 of file `functional_hash.h`.

The documentation for this struct was generated from the following file:

- [functional\\_hash.h](#)

#### 4.554 `std::hash< wstring >` Struct Template Reference

Inherits `std::__hash_base< _Result, _Arg >`.

#### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

#### Public Member Functions

- `size_t operator()` (`const wstring &__s`) `const noexcept`

##### 4.554.1 Detailed Description

```
template<>
struct std::hash< wstring >
```

`std::hash` specialization for `wstring`.

Definition at line 6823 of file `basic_string.h`.

The documentation for this struct was generated from the following file:

- [basic\\_string.h](#)



#### 4.555 `std::hash<::bitset<_Nb>>` Struct Template Reference

Inherits `std::__hash_base<_Result, _Arg>`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (`const ::bitset<_Nb> &__b`) `const noexcept`

##### 4.555.1 Detailed Description

```
template<size_t _Nb>
struct std::hash<::bitset<_Nb>>
```

`std::hash` specialization for `bitset`.

Definition at line 1569 of file `bitset`.

The documentation for this struct was generated from the following file:

- [bitset](#)

#### 4.556 `std::hash<::vector<bool, _Alloc>>` Struct Template Reference

Inherits `std::__hash_base<_Result, _Arg>`.

##### Public Types

- typedef `_Arg` **argument\_type**
- typedef `_Result` **result\_type**

##### Public Member Functions

- `size_t operator()` (`const ::vector<bool, _Alloc> &`) `const noexcept`

#### 4.556.1 Detailed Description

```
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >
```

std::hash specialization for vector<bool>.

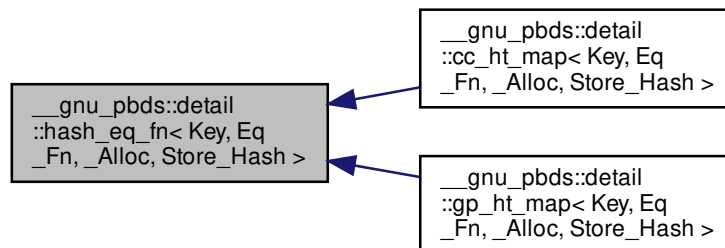
Definition at line 1351 of file stl\_bvector.h.

The documentation for this struct was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

#### 4.557 \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >:



#### 4.557.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >
```

Primary template.

Definition at line 55 of file hash\_eq\_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

4.558 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >` Struct Template Reference

Inherits `Eq_Fn`.

## Public Types

- typedef `Eq_Fn` **eq\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**

## Public Member Functions

- **hash\_eq\_fn** (const `Eq_Fn` &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, key\_const\_reference r\_rhs\_key) const
- void **swap** (const `hash_eq_fn` &other)

## 4.558.1 Detailed Description

```
template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
```

Specialization 1 - The client requests that hash values not be stored.

Definition at line 59 of file `hash_eq_fn.hpp`.

The documentation for this struct was generated from the following file:

- `hash_eq_fn.hpp`

4.559 `__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >` Struct Template Reference

Inherits `Eq_Fn`.

## Public Types

- typedef `Eq_Fn` **eq\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

## Public Member Functions

- **hash\_eq\_fn** (const `Eq_Fn` &r\_eq\_fn)
- bool **operator()** (key\_const\_reference r\_lhs\_key, size\_type lhs\_hash, key\_const\_reference r\_rhs\_key, size\_type rhs\_hash) const
- void **swap** (const `hash_eq_fn` &other)

#### 4.559.1 Detailed Description

```
template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >
```

Specialization 2 - The client requests that hash values be stored.

Definition at line 82 of file hash\_eq\_fn.hpp.

The documentation for this struct was generated from the following file:

- [hash\\_eq\\_fn.hpp](#)

#### 4.560 \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type > Class Template Reference

##### Public Types

- typedef Size\_Type **size\_type**

##### Public Member Functions

- [hash\\_exponential\\_size\\_policy](#) (size\_type start\_size=8, size\_type grow\_factor=2)
- void **swap** ([hash\\_exponential\\_size\\_policy](#)< Size\_Type > &other)

##### Protected Member Functions

- size\_type **get\_nearest\_larger\_size** (size\_type size) const
- size\_type **get\_nearest\_smaller\_size** (size\_type size) const

#### 4.560.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >
```

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2).

Definition at line 413 of file hash\_policy.hpp.

#### 4.560.2 Constructor & Destructor Documentation

#### 4.560.2.1 `hash_exponential_size_policy()`

```
template<typename Size_Type >
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
 size_type start_size = 8,
 size_type grow_factor = 2)
```

Default constructor, or onstructor taking a start\_size, or constructor taking a start size and grow\_factor. The policy will use the sequence of sizes start\_size, start\_size\* grow\_factor, start\_size\* grow\_factor^2, ...

Definition at line 46 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.561 `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`:



#### Public Types

- enum { [external\\_load\\_access](#) }
- typedef Size\_Type **size\_type**

#### Public Member Functions

- [hash\\_load\\_check\\_resize\\_trigger](#) (float load\_min=0.125, float load\_max=0.5)
- `std::pair< float, float >` [get\\_loads](#) () const
- void [set\\_loads](#) (`std::pair< float, float >` load\_pair)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger](#) &other)

## Protected Member Functions

- bool **is\_grow\_needed** (size\_type size, size\_type num\_entries) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (size\_type num\_entries)
- void **notify\_externally\_resized** (size\_type new\_size)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (size\_type num\_entries)
- void **notify\_resized** (size\_type new\_size)

### 4.561.1 Detailed Description

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
```

A resize trigger policy based on a load check. It keeps the load factor between some load factors `load_min` and `load_max`.

Definition at line 175 of file `hash_policy.hpp`.

### 4.561.2 Member Enumeration Documentation

#### 4.561.2.1 anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

#### Enumerator

|                                   |                                                                                                                                                                 |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>external_load_access</code> | Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation. |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 180 of file `hash_policy.hpp`.

#### 4.561.3 Constructor & Destructor Documentation

##### 4.561.3.1 `hash_load_check_resize_trigger()`

```
template<bool External_Load_Access, typename Size_Type >
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_resize_trigger
(
 float load_min = 0.125,
 float load_max = 0.5)
```

Default constructor, or constructor taking `load_min` and `load_max` load factors between which this policy will keep the actual load.

Definition at line 49 of file `hash_policy.hpp`.

#### 4.561.4 Member Function Documentation

##### 4.561.4.1 `get_loads()`

```
template<bool External_Load_Access, typename Size_Type >
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::get_loads () const [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

Definition at line 238 of file `hash_policy.hpp`.

##### 4.561.4.2 `notify_cleared()`

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_cleared () [protected]
```

Notifies the table was cleared.

Definition at line 208 of file `hash_policy.hpp`.

#### 4.561.4.3 notify\_inserted()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
inserted (
 size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is num\_entries.

Definition at line 111 of file hash\_policy.hpp.

#### 4.561.4.4 notify\_resized()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↵
resized (
 size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

Definition at line 153 of file hash\_policy.hpp.

#### 4.561.4.5 set\_loads()

```
template<bool External_Load_Access, typename Size_Type >
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
 std::pair< float, float > load_pair)
```

Sets the loads through a pair of the minimal and maximal loads, respectively.

Definition at line 247 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

### 4.562 \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size > Class Template Reference

#### 4.562.1 Detailed Description

```
template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >
```

Primary template.

Definition at line 50 of file hash\_load\_check\_resize\_trigger\_size\_base.hpp.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)



## 4.563 `__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >` Class Template Reference

### Protected Types

- typedef `Size_Type` **size\_type**

### Protected Member Functions

- `size_type` **get\_size** () const
- void **set\_size** (`size_type` size)
- void **swap** ([hash\\_load\\_check\\_resize\\_trigger\\_size\\_base](#) &other)

### 4.563.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >
```

Specializations.

Definition at line 54 of file `hash_load_check_resize_trigger_size_base.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_load\\_check\\_resize\\_trigger\\_size\\_base.hpp](#)

## 4.564 `__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >` Class Template Reference

### Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Ht::const_pointer` **const\_pointer**
- typedef `_Ht::const_reference` **const\_reference**
- typedef `_Tp` **data\_type**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

## Public Member Functions

- **hash\_map** (size\_type \_\_n)
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_map** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- template<class \_InputIterator >  
**hash\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_map** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_it)
- void **erase** (iterator \_\_f, iterator \_\_l)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- pair< iterator, bool > **insert** (const value\_type &\_\_obj)
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- pair< iterator, bool > **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- \_Tp & **operator[]** (const key\_type &\_\_key)
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** (hash\_map &\_\_hs)

## Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_AI >  
bool **operator==** (const hash\_map< \_K1, \_T1, \_HF, \_EqK, \_AI > &, const hash\_map< \_K1, \_T1, \_HF, \_EqK, \_AI > &)

## 4.565 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference 2671

---

### 4.564.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_map<_Key, _Tp, _HashFn, _EqualKey, _Alloc>
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation__style.html)

Definition at line 83 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

## 4.565 `__gnu_cxx::hash_multimap<_Key, _Tp, _HashFn, _EqualKey, _Alloc>` Class Template Reference

### Public Types

- typedef `_Ht::allocator_type` **allocator\_type**
- typedef `_Ht::const_iterator` **const\_iterator**
- typedef `_Ht::const_pointer` **const\_pointer**
- typedef `_Ht::const_reference` **const\_reference**
- typedef `_Tp` **data\_type**
- typedef `_Ht::difference_type` **difference\_type**
- typedef `_Ht::hasher` **hasher**
- typedef `_Ht::iterator` **iterator**
- typedef `_Ht::key_equal` **key\_equal**
- typedef `_Ht::key_type` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Ht::pointer` **pointer**
- typedef `_Ht::reference` **reference**
- typedef `_Ht::size_type` **size\_type**
- typedef `_Ht::value_type` **value\_type**

## Public Member Functions

- **hash\_multimap** (size\_type \_\_n)
- **hash\_multimap** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- template<class \_InputIterator >  
**hash\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_multimap** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** ()
- const\_iterator **begin** () const
- size\_type **bucket\_count** () const
- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_it)
- void **erase** (iterator \_\_f, iterator \_\_l)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- iterator **insert** (const value\_type &\_\_obj)
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- iterator **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** (hash\_multimap &\_\_hs)

## Friends

- template<class \_K1, class \_T1, class \_HF, class \_EqK, class \_AI >  
bool **operator==** (const hash\_multimap< \_K1, \_T1, \_HF, \_EqK, \_AI > &, const hash\_multimap< \_K1, \_T1, \_HF, \_EqK, \_AI > &)

## 4.565.1 Detailed Description

```
template<class _Key, class _Tp, class _HashFcn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_multimap<_Key, _Tp, _HashFcn, _EqualKey, _Alloc>
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 296 of file `hash_map`.

The documentation for this class was generated from the following file:

- [hash\\_map](#)

4.566 `__gnu_cxx::hash_multiset<_Value, _HashFcn, _EqualKey, _Alloc>` Class Template Reference

## Public Types

- `typedef _Ht::allocator_type allocator_type`
- `typedef _Ht::const_iterator const_iterator`
- `typedef _Alloc::const_pointer const_pointer`
- `typedef _Alloc::const_reference const_reference`
- `typedef _Ht::difference_type difference_type`
- `typedef _Ht::hasher hasher`
- `typedef _Ht::const_iterator iterator`
- `typedef _Ht::key_equal key_equal`
- `typedef _Ht::key_type key_type`
- `typedef _Alloc::pointer pointer`
- `typedef _Alloc::reference reference`
- `typedef _Ht::size_type size_type`
- `typedef _Ht::value_type value_type`

## Public Member Functions

- `hash_multiset (size_type __n)`
- `hash_multiset (size_type __n, const hasher &__hf)`
- `hash_multiset (size_type __n, const hasher &__hf, const key_equal &__eq, const allocator_type &__a=allocator_type())`
- `template<class _InputIterator> hash_multiset (_InputIterator __f, _InputIterator __l)`
- `template<class _InputIterator> hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)`
- `template<class _InputIterator> hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)`

- `template<class _InputIterator >`  
`hash_multiset` (`_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eq,`  
`const allocator_type &__a=allocator_type()`)
- iterator `begin` () const
- size\_type `bucket_count` () const
- void `clear` ()
- size\_type `count` (const key\_type &\_\_key) const
- size\_type `elems_in_bucket` (size\_type \_\_n) const
- bool `empty` () const
- iterator `end` () const
- `pair`< iterator, iterator > `equal_range` (const key\_type &\_\_key) const
- size\_type `erase` (const key\_type &\_\_key)
- void `erase` (iterator \_\_it)
- void `erase` (iterator \_\_f, iterator \_\_l)
- iterator `find` (const key\_type &\_\_key) const
- allocator\_type `get_allocator` () const
- hasher `hash_func` () const
- iterator `insert` (const value\_type &\_\_obj)
- `template<class _InputIterator >`  
`void insert` (`_InputIterator __f, _InputIterator __l`)
- iterator `insert_noresize` (const value\_type &\_\_obj)
- key\_equal `key_eq` () const
- size\_type `max_bucket_count` () const
- size\_type `max_size` () const
- void `resize` (size\_type \_\_hint)
- size\_type `size` () const
- void `swap` (`hash_multiset` &hs)

#### Friends

- `template<class _Val, class _HF, class _EqK, class _AI >`  
`bool operator==` (const `hash_multiset`< `_Val, _HF, _EqK, _AI` > &, const `hash_multiset`< `_Val, _HF, _EqK, _AI`  
 > &)

#### 4.566.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 287 of file `hash_set`.

The documentation for this class was generated from the following file:

- `hash_set`

## 4.567 `__gnu_pbds::hash_prime_size_policy` Class Reference

### Public Types

- typedef std::size\_t `size_type`

### Public Member Functions

- `hash_prime_size_policy` (`size_type` start\_size=8)
- void `swap` (`hash_prime_size_policy` &other)

### Protected Member Functions

- `size_type` `get_nearest_larger_size` (`size_type` size) const
- `size_type` `get_nearest_smaller_size` (`size_type` size) const

#### 4.567.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

Definition at line 450 of file `hash_policy.hpp`.

#### 4.567.2 Member Typedef Documentation

##### 4.567.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

Definition at line 454 of file `hash_policy.hpp`.

#### 4.567.3 Constructor & Destructor Documentation

#### 4.567.3.1 hash\_prime\_size\_policy()

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (
 size_type start_size = 8) [inline]
```

Default constructor, or onstructor taking a start\_size The policy will use the sequence of sizes approximately start\_size, start\_size\* 2, start\_size\* 2^2, ...

Definition at line 129 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

#### 4.568 \_\_gnu\_cxx::hash\_set< \_Value, \_HashFcn, \_EqualKey, \_Alloc > Class Template Reference

##### Public Types

- typedef \_Ht::allocator\_type **allocator\_type**
- typedef \_Ht::const\_iterator **const\_iterator**
- typedef \_Alloc\_traits::const\_pointer **const\_pointer**
- typedef \_Alloc\_traits::const\_reference **const\_reference**
- typedef \_Ht::difference\_type **difference\_type**
- typedef \_Ht::hasher **hasher**
- typedef \_Ht::const\_iterator **iterator**
- typedef \_Ht::key\_equal **key\_equal**
- typedef \_Ht::key\_type **key\_type**
- typedef \_Alloc\_traits::pointer **pointer**
- typedef \_Alloc\_traits::reference **reference**
- typedef \_Ht::size\_type **size\_type**
- typedef \_Ht::value\_type **value\_type**

##### Public Member Functions

- **hash\_set** (size\_type \_\_n)
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf)
- **hash\_set** (size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf)
- template<class \_InputIterator >  
**hash\_set** (\_InputIterator \_\_f, \_InputIterator \_\_l, size\_type \_\_n, const hasher &\_\_hf, const key\_equal &\_\_eq, const allocator\_type &\_\_a=allocator\_type())
- iterator **begin** () const
- size\_type **bucket\_count** () const



- void **clear** ()
- size\_type **count** (const key\_type &\_\_key) const
- size\_type **elems\_in\_bucket** (size\_type \_\_n) const
- bool **empty** () const
- iterator **end** () const
- [pair](#)< iterator, iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- void **erase** (iterator \_\_it)
- void **erase** (iterator \_\_f, iterator \_\_l)
- iterator **find** (const key\_type &\_\_key) const
- allocator\_type **get\_allocator** () const
- hasher **hash\_funct** () const
- [pair](#)< iterator, bool > **insert** (const value\_type &\_\_obj)
- template<class \_InputIterator >  
void **insert** (\_InputIterator \_\_f, \_InputIterator \_\_l)
- [pair](#)< iterator, bool > **insert\_noresize** (const value\_type &\_\_obj)
- key\_equal **key\_eq** () const
- size\_type **max\_bucket\_count** () const
- size\_type **max\_size** () const
- void **resize** (size\_type \_\_hint)
- size\_type **size** () const
- void **swap** ([hash\\_set](#) &\_\_hs)

#### Friends

- template<class \_Val, class \_HF, class \_EqK, class \_Al >  
bool **operator==** (const [hash\\_set](#)< \_Val, \_HF, \_EqK, \_Al > &, const [hash\\_set](#)< \_Val, \_HF, \_EqK, \_Al > &)

#### 4.568.1 Detailed Description

```
template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_set<_Value, _HashFcn, _EqualKey, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 84 of file `hash_set`.

The documentation for this class was generated from the following file:

- [hash\\_set](#)

#### 4.569 `__gnu_pbds::hash_standard_resize_policy`< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` > Class Template Reference

Inherits `Size_Policy`, and `Trigger_Policy`.

##### Public Types

- enum { **external\_size\_access** }
- typedef `Size_Policy` **size\_policy**
- typedef `Size_Type` **size\_type**
- typedef `Trigger_Policy` **trigger\_policy**

##### Public Member Functions

- [hash\\_standard\\_resize\\_policy](#) ()
- [hash\\_standard\\_resize\\_policy](#) (const `Size_Policy` &`r_size_policy`)
- [hash\\_standard\\_resize\\_policy](#) (const `Size_Policy` &`r_size_policy`, const `Trigger_Policy` &`r_trigger_policy`)
- `size_type` [get\\_actual\\_size](#) () const
- `Size_Policy` & [get\\_size\\_policy](#) ()
- const `Size_Policy` & [get\\_size\\_policy](#) () const
- `Trigger_Policy` & [get\\_trigger\\_policy](#) ()
- const `Trigger_Policy` & [get\\_trigger\\_policy](#) () const
- void [resize](#) (`size_type` `suggested_new_size`)
- void **swap** ([hash\\_standard\\_resize\\_policy](#)< `Size_Policy`, `Trigger_Policy`, `External_Size_Access`, `Size_Type` > &`other`)

##### Protected Member Functions

- `size_type` [get\\_new\\_size](#) (`size_type` `size`, `size_type` `num_used_e`) const
- bool **is\_resize\_needed** () const
- void **notify\_cleared** ()
- void **notify\_erase\_search\_collision** ()
- void **notify\_erase\_search\_end** ()
- void **notify\_erase\_search\_start** ()
- void **notify\_erased** (`size_type` `num_e`)
- void **notify\_find\_search\_collision** ()
- void **notify\_find\_search\_end** ()
- void **notify\_find\_search\_start** ()
- void **notify\_insert\_search\_collision** ()
- void **notify\_insert\_search\_end** ()
- void **notify\_insert\_search\_start** ()
- void **notify\_inserted** (`size_type` `num_e`)
- void **notify\_resized** (`size_type` `new_size`)

#### 4.569.1 Detailed Description

```
template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_↵
trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
```

A resize policy which delegates operations to size and trigger policies.

Definition at line 489 of file `hash_policy.hpp`.

#### 4.569.2 Constructor & Destructor Documentation

##### 4.569.2.1 `hash_standard_resize_policy()` [1/3]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_↵
_Type >::hash_standard_resize_policy ()
```

Default constructor.

Definition at line 46 of file `hash_policy.hpp`.

##### 4.569.2.2 `hash_standard_resize_policy()` [2/3]

```
template<typename Size_Policy, typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_↵
_Type >::hash_standard_resize_policy (
 const Size_Policy & r_size_policy)
```

constructor taking some policies `r_size_policy` will be copied by the `Size_Policy` object of this object.

Definition at line 52 of file `hash_policy.hpp`.

##### 4.569.2.3 `hash_standard_resize_policy()` [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size_↵
_Type >
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_↵
_Type >::hash_standard_resize_policy (
 const Size_Policy & r_size_policy,
 const Trigger_Policy & r_trigger_policy)
```

constructor taking some policies. `r_size_policy` will be copied by the `Size_Policy` object of this object. `r_trigger_policy` will be copied by the `Trigger_Policy` object of this object.

Definition at line 58 of file `hash_policy.hpp`.

### 4.569.3 Member Function Documentation

#### 4.569.3.1 `get_actual_size()`

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↵
Access, Size_Type >::get_actual_size () const [inline]
```

Returns the actual size of the container.

Definition at line 179 of file `hash_policy.hpp`.

#### 4.569.3.2 `get_new_size()`

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↵
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↵
Access, Size_Type >::get_new_size (
 size_type size,
 size_type num_used_e) const [protected]
```

Queries what the new size should be, when the container is resized naturally. The current `__size` of the container is `size`, and the number of used entries within the container is `num_used_e`.

Definition at line 160 of file `hash_policy.hpp`.

#### 4.569.3.3 `get_size_policy()` [1/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
Size_Access, Size_Type >::get_size_policy ()
```

Access to the `Size_Policy` object used.

Definition at line 244 of file `hash_policy.hpp`.

#### 4.569.3.4 `get_size_policy()` [2/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
_Size_Access, Size_Type >::get_size_policy () const
```

Const access to the `Size_Policy` object used.

Definition at line 250 of file `hash_policy.hpp`.

#### 4.569.3.5 `get_trigger_policy()` [1/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↵
_Size_Access, Size_Type >::get_trigger_policy ()
```

Access to the `Trigger_Policy` object used.

Definition at line 232 of file `hash_policy.hpp`.

#### 4.569.3.6 `get_trigger_policy()` [2/2]

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy () const
```

Access to the `Trigger_Policy` object used.

Definition at line 238 of file `hash_policy.hpp`.

#### 4.569.3.7 `resize()`

```
template<typename Size_Policy , typename Trigger_Policy , bool External_Size_Access, typename
Size_Type >
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
 size_type suggested_new_size)
```

Resizes the container to `suggested_new_size`, a suggested size (the actual size will be determined by the `Size_Policy` object).

Definition at line 188 of file `hash_policy.hpp`.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 4.570 `std::thread::id` Class Reference

### Public Member Functions

- `id` (`native_handle_type __id`)

### Friends

- class `hash< id >`
- bool `operator<` (`id __x`, `id __y`) noexcept
- template<class `_CharT`, class `_Traits` >  
`basic_ostream< _CharT, _Traits > & operator<<` (`basic_ostream< _CharT, _Traits > &__out`, `id __id`)
- bool `operator==` (`id __x`, `id __y`) noexcept
- class `thread`

### 4.570.1 Detailed Description

`thread::id`

Definition at line 88 of file `thread`.

The documentation for this class was generated from the following file:

- `thread`

## 4.571 `std::locale::id` Class Reference

### Public Member Functions

- `id` ()
- `size_t _M_id` () const throw ()

### Friends

- template<typename `_Facet` >  
bool `has_facet` (const `locale` &) throw ()
- class `locale`
- class `locale::_Impl`
- template<typename `_Facet` >  
const `_Facet` & `use_facet` (const `locale` &)

#### 4.571.1 Detailed Description

Facet ID class.

The ID class provides facets with an index used to identify them. Every facet class must define a public static member `locale::id`, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The `locale::id` ensures that each class type gets a unique identifier.

Definition at line 485 of file `locale_classes.h`.

#### 4.571.2 Constructor & Destructor Documentation

##### 4.571.2.1 id()

```
std::locale::id::id () [inline]
```

Constructor.

Definition at line 516 of file `locale_classes.h`.

#### 4.571.3 Friends And Related Function Documentation

##### 4.571.3.1 has\_facet

```
template<typename _Facet >
bool has_facet (
 const locale &) throw () [friend]
```

Test for the presence of a facet.

`has_facet` tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

##### Template Parameters

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

##### Parameters

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

**Returns**

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file `locale_classes.tcc`.

**4.571.3.2 use\_facet**

```
template<typename _Facet >
const _Facet& use_facet (
 const locale &) [friend]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

**Template Parameters**

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

**Parameters**

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

**Returns**

Reference to facet of type `Facet`.

**Exceptions**

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

Definition at line 132 of file `locale_classes.tcc`.

The documentation for this class was generated from the following file:

- [locale\\_classes.h](#)

**4.572 std::experimental::fundamentals\_v1::in\_place\_t Struct Reference****4.572.1 Detailed Description**

Tag type for in-place construction.



Definition at line 74 of file `experimental/optional`.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

## 4.573 `std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >` Class Template Reference

### Public Types

- typedef `_UIntType` [result\\_type](#)

### Public Member Functions

- [independent\\_bits\\_engine](#) ()
- [independent\\_bits\\_engine](#) (const `_RandomNumberEngine` &\_\_rng)
- [independent\\_bits\\_engine](#) (`_RandomNumberEngine` &&\_\_rng)
- [independent\\_bits\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq`, typename `=_If_seed_seq<_Sseq>>`  
[independent\\_bits\\_engine](#) (`_Sseq` &\_\_q)
- const `_RandomNumberEngine` & [base](#) () const noexcept
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator](#)() ()
- void [seed](#) ()
- void [seed](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq` >  
`_If_seed_seq<_Sseq >` [seed](#) (`_Sseq` &\_\_q)

### Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

### Friends

- bool [operator==](#) (const [independent\\_bits\\_engine](#) &\_\_lhs, const [independent\\_bits\\_engine](#) &\_\_rhs)
- template<typename `_CharT`, typename `_Traits` >  
[std::basic\\_istream](#)< `_CharT`, `_Traits` > & [operator>>](#) ([std::basic\\_istream](#)< `_CharT`, `_Traits` > &\_\_is,  
[std::independent\\_bits\\_engine](#)< `_RandomNumberEngine`, \_\_w, `_UIntType` > &\_\_x)

#### 4.573.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits `__w`.

Definition at line 1105 of file `random.h`.

### 4.573.2 Member Typedef Documentation

#### 4.573.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type
```

The type of the generated random value.

Definition at line 1118 of file random.h.

### 4.573.3 Constructor & Destructor Documentation

#### 4.573.3.1 independent\_bits\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ()
[inline]
```

Constructs a default `independent_bits_engine` engine.

The underlying engine is default constructed as well.

Definition at line 1125 of file random.h.

#### 4.573.3.2 independent\_bits\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
 const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a `independent_bits_engine` engine.

Copies an existing base class random number generator.

##### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1135 of file random.h.

## 4.573.3.3 independent\_bits\_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
 _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a independent\_bits\_engine engine.

Copies an existing base class random number generator.

## Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1145 of file random.h.

## 4.573.3.4 independent\_bits\_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
 result_type __s) [inline], [explicit]
```

Seed constructs a independent\_bits\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

Definition at line 1155 of file random.h.

## 4.573.3.5 independent\_bits\_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::independent_bits_engine<_RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
 _Sseq & __q) [inline], [explicit]
```

Generator construct a independent\_bits\_engine engine.

#### Parameters

|                       |                  |
|-----------------------|------------------|
| <code>↔<br/>_q</code> | A seed sequence. |
|-----------------------|------------------|

Definition at line 1165 of file random.h.

### 4.573.4 Member Function Documentation

#### 4.573.4.1 base()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine& std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >↔
::base () const [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1200 of file random.h.

#### 4.573.4.2 discard()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 1221 of file random.h.

#### 4.573.4.3 max()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>::max () [inline], [static]
```

Gets the maximum value in the generated random number range.

Definition at line 1214 of file random.h.

#### 4.573.4.4 `min()`

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>::min () [inline], [static]
```

Gets the minimum value in the generated random number range.

Definition at line 1207 of file `random.h`.

#### 4.573.4.5 `operator()()`

```
template<typename _RandomNumberEngine , size_t __w, typename _UIntType >
independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()
```

Gets the next value in the generated random number sequence.

Definition at line 737 of file `bits/random.tcc`.

References `std::__lg()`, `std::numeric_limits<_Tp>::max()`, and `std::numeric_limits<_Tp>::min()`.

#### 4.573.4.6 `seed()` [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed () [inline]
```

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1174 of file `random.h`.

#### 4.573.4.7 `seed()` [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
 result_type __s) [inline]
```

Reseeds the `independent_bits_engine` object with the default seed for the underlying base class generator engine.

Definition at line 1182 of file `random.h`.

#### 4.573.4.8 `seed()` [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq >
_If_seed_seq<_Sseq> std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
 _Sseq & __q) [inline]
```

Reseeds the `independent_bits_engine` object with the given seed sequence.

**Parameters**

|                 |                            |
|-----------------|----------------------------|
| <code>_↵</code> | A seed generator function. |
| <code>_q</code> |                            |

Definition at line 1192 of file random.h.

**4.573.5 Friends And Related Function Documentation****4.573.5.1 operator==**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
 const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
 const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs) [friend]
```

Compares two independent\_bits\_engine random number generator objects of the same type for equality.

**Parameters**

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| <code>__lhs</code> | A independent_bits_engine random number generator object.       |
| <code>__rhs</code> | Another independent_bits_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.



Definition at line 1246 of file random.h.

**4.573.5.2 operator>>**

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

## Parameters

|                                                                                                                       |                                                           |
|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| <a href="#"></a><br><code>_is</code> | An input stream.                                          |
| <a href="#"></a><br><code>_x</code>  | A independent_bits_engine random number generator engine. |

## Returns

The input stream with the state of `__x` extracted or in an error state.

Definition at line 1264 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.574 std::indirect\_array&lt;\_Tp&gt; Class Template Reference

## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- [indirect\\_array](#) (const [indirect\\_array](#) &)
- void [operator%=\(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator%=\(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator&=\(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator&=\(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator\\*=\(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator\\*=\(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator+=\(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator+=\(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator-= \(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator-= \(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator/= \(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator/= \(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const
- void [operator<=<= \(const valarray<\\_Tp> &\)](#) const
- template<class `_Dom` >  
void [operator<<=<= \(const \\_Expr<\\_Dom, \\_Tp> &\)](#) const

- `indirect_array` & `operator=` (const `indirect_array` &)
- void `operator=` (const `valarray`<\_Tp> &) const
- void `operator=` (const \_Tp &) const
- template<class \_Dom >  
void `operator=` (const \_Expr<\_Dom, \_Tp> &) const
- void `operator>>=` (const `valarray`<\_Tp> &) const
- template<class \_Dom >  
void `operator>>=` (const \_Expr<\_Dom, \_Tp> &) const
- void `operator^=` (const `valarray`<\_Tp> &) const
- template<class \_Dom >  
void `operator^=` (const \_Expr<\_Dom, \_Tp> &) const
- void `operator|=` (const `valarray`<\_Tp> &) const
- template<class \_Dom >  
void `operator|=` (const \_Expr<\_Dom, \_Tp> &) const

#### Friends

- class `gslice_array`<\_Tp>
- class `valarray`<\_Tp>

#### 4.574.1 Detailed Description

```
template<class _Tp>
class std::indirect_array<_Tp>
```

Reference to arbitrary subset of an array.

An `indirect_array` is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an `indirect_array` is to call `operator[]`(`valarray`<size\_t>) on a `valarray`. The returned `indirect_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if an `indirect_array` is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have `array[0]==3`, `array[2]==2`, and `array[4]==1`.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

Definition at line 95 of file `valarray`.

The documentation for this class was generated from the following files:

- `valarray`
- `indirect_array.h`



## 4.575 std::initializer\_list&lt;\_E&gt; Class Template Reference

## Public Types

- typedef const \_E \* **const\_iterator**
- typedef const \_E & **const\_reference**
- typedef const \_E \* **iterator**
- typedef const \_E & **reference**
- typedef size\_t **size\_type**
- typedef \_E **value\_type**

## Public Member Functions

- constexpr const\_iterator **begin** () const noexcept
- constexpr const\_iterator **end** () const noexcept
- constexpr size\_type **size** () const noexcept

## Related Functions

(Note that these are not member functions.)

- template<class \_Tp >  
constexpr const \_Tp \* [begin](#) (initializer\_list<\_Tp> \_\_ils) noexcept
- template<class \_Tp >  
constexpr const \_Tp \* [end](#) (initializer\_list<\_Tp> \_\_ils) noexcept

## 4.575.1 Detailed Description

```
template<class _E>
class std::initializer_list<_E>
```

initializer\_list

Definition at line 47 of file initializer\_list.

## 4.575.2 Friends And Related Function Documentation

## 4.575.2.1 begin()

```
template<class _Tp >
constexpr const _Tp * begin (
 initializer_list<_Tp> __ils) [related]
```

Return an iterator pointing to the first element of the initializer\_list.

**Parameters**

|                                  |                   |
|----------------------------------|-------------------|
| $\leftrightarrow$<br><i>_ils</i> | Initializer list. |
|----------------------------------|-------------------|

Definition at line 90 of file `initializer_list`.

**4.575.2.2 `end()`**

```
template<class _Tp >
constexpr const _Tp * end (
 initializer_list< _Tp > __ils) [related]
```

Return an iterator pointing to one past the last element of the `initializer_list`.

**Parameters**

|                                  |                   |
|----------------------------------|-------------------|
| $\leftrightarrow$<br><i>_ils</i> | Initializer list. |
|----------------------------------|-------------------|

Definition at line 101 of file `initializer_list`.

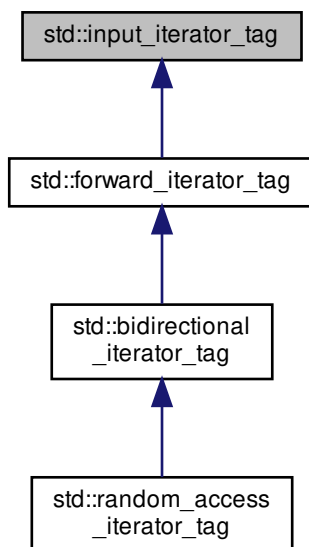
Referenced by `std::forward_list<_Tp, _Alloc>::resize()`.

The documentation for this class was generated from the following file:

- [initializer\\_list](#)

## 4.576 std::input\_iterator\_tag Struct Reference

Inheritance diagram for std::input\_iterator\_tag:



## 4.576.1 Detailed Description

Marking input iterators.

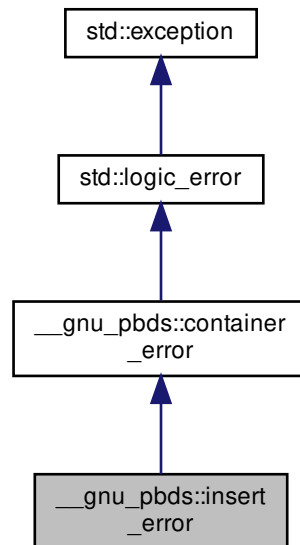
Definition at line 93 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.577 `__gnu_pbds::insert_error` Struct Reference

Inheritance diagram for `__gnu_pbds::insert_error`:



##### Public Member Functions

- virtual const char \* [what](#) () const noexcept

##### 4.577.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unavailable, in which case the `allocator_type`'s exception will be thrown).

Definition at line 66 of file `exception.hpp`.

##### 4.577.2 Member Function Documentation

4.577.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

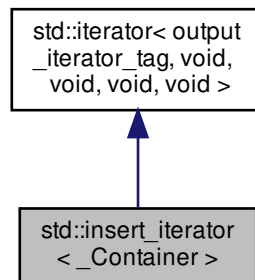
Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

4.578 `std::insert_iterator< _Container >` Class Template Reference

Inheritance diagram for `std::insert_iterator< _Container >`:



## Public Types

- typedef `_Container` [container\\_type](#)
- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

## Public Member Functions

- constexpr [insert\\_iterator](#) (`_Container &__x, _Iter __i`)
- constexpr [insert\\_iterator](#) & [operator\\*](#) ()
- constexpr [insert\\_iterator](#) & [operator++](#) ()
- constexpr [insert\\_iterator](#) & [operator++](#) (int)
- constexpr [insert\\_iterator](#) & [operator=](#) (const typename `_Container::value_type` &\_\_value)
- constexpr [insert\\_iterator](#) & [operator=](#) (typename `_Container::value_type` &&\_\_value)

## Protected Attributes

- `_Container * container`
- `_Iter iter`

### 4.578.1 Detailed Description

```
template<typename _Container>
class std::insert_iterator< _Container >
```

Turns assignment into insertion.

These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.

(Sequences will actually insert a *copy* of the value before the iterator's position.)

Tip: Using the inserter function to create these iterators can save typing.

Definition at line 806 of file `bits/stl_iterator.h`.

### 4.578.2 Member Typedef Documentation

#### 4.578.2.1 `container_type`

```
template<typename _Container >
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

Definition at line 825 of file `bits/stl_iterator.h`.

#### 4.578.2.2 `difference_type`

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file `stl_iterator_base_types.h`.

#### 4.578.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.578.2.4 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file stl\_iterator\_base\_types.h.

#### 4.578.2.5 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file stl\_iterator\_base\_types.h.

#### 4.578.2.6 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file stl\_iterator\_base\_types.h.

### 4.578.3 Constructor & Destructor Documentation

#### 4.578.3.1 insert\_iterator()

```
template<typename _Container >
constexpr std::insert_iterator< _Container >::insert_iterator (
 _Container & __x,
 _Iter __i) [inline]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).

Definition at line 838 of file bits/stl\_iterator.h.

#### 4.578.4 Member Function Documentation

##### 4.578.4.1 `operator*()`

```
template<typename _Container >
constexpr insert_iterator& std::insert_iterator< _Container >::operator* () [inline]
```

Simply returns `*this`.

Definition at line 895 of file `bits/stl_iterator.h`.

##### 4.578.4.2 `operator++()` [1/2]

```
template<typename _Container >
constexpr insert_iterator& std::insert_iterator< _Container >::operator++ () [inline]
```

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 901 of file `bits/stl_iterator.h`.

##### 4.578.4.3 `operator++()` [2/2]

```
template<typename _Container >
constexpr insert_iterator& std::insert_iterator< _Container >::operator++ (
 int) [inline]
```

Simply returns `*this`. (This iterator does not *move*.)

Definition at line 907 of file `bits/stl_iterator.h`.

##### 4.578.4.4 `operator=()`

```
template<typename _Container >
constexpr insert_iterator& std::insert_iterator< _Container >::operator= (
 const typename _Container::value_type & __value) [inline]
```

###### Parameters

|                      |                                                                                                                                                       |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__value</code> | An instance of whatever type <code>container_type::const_reference</code> is; presumably a reference-to-const T for <code>container&lt;T&gt;</code> . |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|



### Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.

The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z
insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;
// vector v contains A, 1, 2, 3, and Z
```

Definition at line 875 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

## 4.579 `std::integer_sequence<_Tp, _Idx>` Struct Template Reference

### Public Types

- typedef `_Tp` **value\_type**

### Static Public Member Functions

- static constexpr `size_t` **size** () noexcept

#### 4.579.1 Detailed Description

```
template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence<_Tp, _Idx>
```

Class template `integer_sequence`.

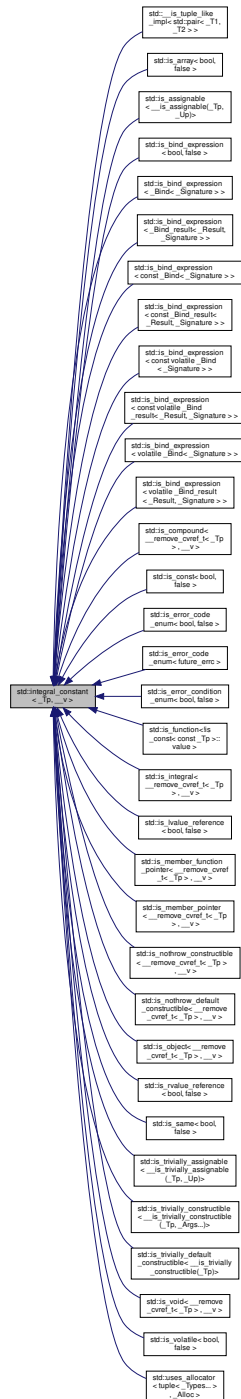
Definition at line 326 of file `utility`.

The documentation for this struct was generated from the following file:

- [utility](#)

## 4.580 std::integral\_constant< \_Tp, \_\_v > Struct Template Reference

Inheritance diagram for std::integral\_constant< \_Tp, \_\_v >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.580.1 Detailed Description

```
template<typename _Tp, _Tp __v>
struct std::integral_constant< _Tp, __v >
```

integral\_constant

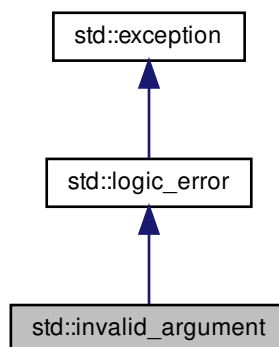
Definition at line 57 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.581 std::invalid\_argument Class Reference

Inheritance diagram for std::invalid\_argument:



## Public Member Functions

- **invalid\_argument** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **invalid\_argument** (const char \*) \_GLIBCXX\_TXN\_SAFE
- **invalid\_argument** (const [invalid\\_argument](#) &)=default
- **invalid\_argument** ([invalid\\_argument](#) &&)=default
- [invalid\\_argument](#) & **operator=** (const [invalid\\_argument](#) &)=default
- [invalid\\_argument](#) & **operator=** ([invalid\\_argument](#) &&)=default
- virtual const char \* [what](#) () const noexcept

### 4.581.1 Detailed Description

Thrown to report invalid arguments to functions.

Definition at line 168 of file stdexcept.

### 4.581.2 Member Function Documentation

#### 4.581.2.1 what()

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

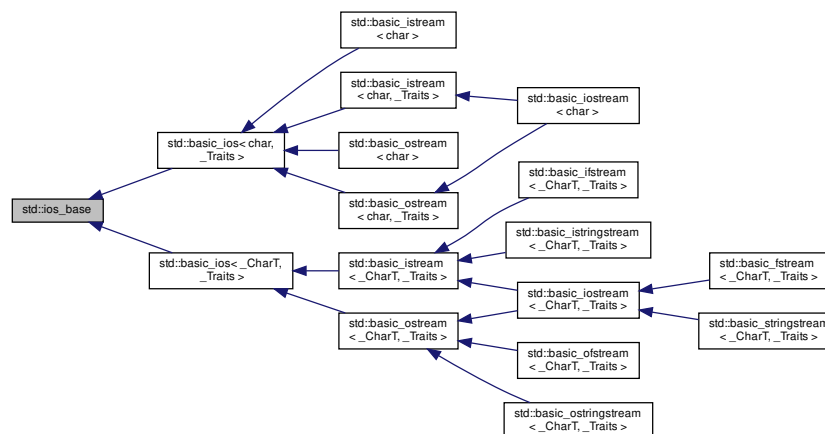
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.582 std::ios\_base Class Reference

Inheritance diagram for std::ios\_base:



## Classes

- class [failure](#)

## Public Types

- typedef int io\_state [\\_GLIBCXX\\_DEPRECATED\\_SUGGEST\("std::iostate"\)](#)
- typedef int open\_mode [\\_GLIBCXX\\_DEPRECATED\\_SUGGEST\("std::openmode"\)](#)
- typedef int seek\_dir [\\_GLIBCXX\\_DEPRECATED\\_SUGGEST\("std::seekdir"\)](#)
- typedef [std::streampos](#) streampos [\\_GLIBCXX\\_DEPRECATED\\_SUGGEST\("std::streampos"\)](#)
- typedef [std::streamoff](#) streamoff [\\_GLIBCXX\\_DEPRECATED\\_SUGGEST\("std::streamoff"\)](#)
- enum [event](#) { [erase\\_event](#), [imbue\\_event](#), [copyfmt\\_event](#) }
- typedef void(\* [event\\_callback](#)) ([event](#) \_\_e, [ios\\_base](#) &\_\_b, int \_\_i)
- typedef \_ios\_Fmtflags [fmtflags](#)
- typedef \_ios\_istate [iostate](#)
- typedef \_ios\_Openmode [openmode](#)
- typedef \_ios\_Seekdir [seekdir](#)

## Public Member Functions

- [ios\\_base](#) (const [ios\\_base](#) &)=delete
- virtual [~ios\\_base](#) ()
- const [locale](#) & [\\_M\\_getloc](#) () const
- [fmtflags](#) [flags](#) () const
- [fmtflags](#) [flags](#) ([fmtflags](#) \_\_fmtfl)
- [locale](#) [getloc](#) () const
- [locale](#) [imbue](#) (const [locale](#) &\_\_loc) throw ()
- long & [iword](#) (int \_\_ix)
- [ios\\_base](#) & [operator=](#) (const [ios\\_base](#) &)=delete
- [streamsize](#) [precision](#) () const
- [streamsize](#) [precision](#) ([streamsize](#) \_\_prec)
- void \*& [pword](#) (int \_\_ix)
- void [register\\_callback](#) ([event\\_callback](#) \_\_fn, int \_\_index)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl)
- [fmtflags](#) [setf](#) ([fmtflags](#) \_\_fmtfl, [fmtflags](#) \_\_mask)
- void [unsetf](#) ([fmtflags](#) \_\_mask)
- [streamsize](#) [width](#) () const
- [streamsize](#) [width](#) ([streamsize](#) \_\_wide)

## Static Public Member Functions

- static bool [sync\\_with\\_stdio](#) (bool \_\_sync=true)
- static int [xalloc](#) () throw ()

### Static Public Attributes

- static const [fmtflags](#) [adjustfield](#)
- static const [openmode](#) [app](#)
- static const [openmode](#) [ate](#)
- static const [iosstate](#) [badbit](#)
- static const [fmtflags](#) [basefield](#)
- static const [seekdir](#) [beg](#)
- static const [openmode](#) [binary](#)
- static const [fmtflags](#) [boolalpha](#)
- static const [seekdir](#) [cur](#)
- static const [fmtflags](#) [dec](#)
- static const [seekdir](#) [end](#)
- static const [iosstate](#) [eofbit](#)
- static const [iosstate](#) [failbit](#)
- static const [fmtflags](#) [fixed](#)
- static const [fmtflags](#) [floatfield](#)
- static const [iosstate](#) [goodbit](#)
- static const [fmtflags](#) [hex](#)
- static const [openmode](#) [in](#)
- static const [fmtflags](#) [internal](#)
- static const [fmtflags](#) [left](#)
- static const [fmtflags](#) [oct](#)
- static const [openmode](#) [out](#)
- static const [fmtflags](#) [right](#)
- static const [fmtflags](#) [scientific](#)
- static const [fmtflags](#) [showbase](#)
- static const [fmtflags](#) [showpoint](#)
- static const [fmtflags](#) [showpos](#)
- static const [fmtflags](#) [skipws](#)
- static const [openmode](#) [trunc](#)
- static const [fmtflags](#) [unitbuf](#)
- static const [fmtflags](#) [uppercase](#)

### Protected Types

- enum { [\\_S\\_local\\_word\\_size](#) }

### Protected Member Functions

- void [\\_M\\_call\\_callbacks](#) ([event](#) \_\_ev) throw ()
- void [\\_M\\_dispose\\_callbacks](#) (void) throw ()
- [\\_Words](#) & [\\_M\\_grow\\_words](#) (int \_\_index, bool \_\_iword)
- void [\\_M\\_init](#) () throw ()
- void [\\_M\\_move](#) ([ios\\_base](#) &) noexcept
- void [\\_M\\_swap](#) ([ios\\_base](#) &\_\_rhs) noexcept

## Protected Attributes

- `_Callback_list * _M_callbacks`
- `iostate _M_exception`
- `fmtflags _M_flags`
- `locale _M_ios_locale`
- `_Words _M_local_word [ _S_local_word_size]`
- `streamsize _M_precision`
- `iostate _M_streambuf_state`
- `streamsize _M_width`
- `_Words * _M_word`
- `int _M_word_size`
- `_Words _M_word_zero`

## 4.582.1 Detailed Description

The base of the I/O class hierarchy.

This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

Definition at line 228 of file `ios_base.h`.

## 4.582.2 Member Typedef Documentation

## 4.582.2.1 event\_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

## Parameters

|                  |                                                        |
|------------------|--------------------------------------------------------|
| <code>__e</code> | One of the members of the event enum.                  |
| <code>__b</code> | Reference to the <code>ios_base</code> object.         |
| <code>__i</code> | The integer provided when the callback was registered. |

Event callbacks are user defined functions that get called during several `ios_base` and `basic_ios` functions, specifically `imbue()`, `copyfmt()`, and `~ios()`.

Definition at line 529 of file `ios_base.h`.

#### 4.582.2.2 fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags
```

This is a bitmask type.

`_Ios_Fmtflags` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `fmtflags` are:

- `boolalpha`
- `dec`
- `fixed`
- `hex`
- `internal`
- `left`
- `oct`
- `right`
- `scientific`
- `showbase`
- `showpoint`
- `showpos`
- `skipws`
- `unitbuf`
- `uppercase`
- `adjustfield`
- `basefield`
- `floatfield`

Definition at line 341 of file `ios_base.h`.

#### 4.582.2.3 iostate

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.

`_Ios_Iostate` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `iostate` are:

- `badbit`
- `eofbit`
- `failbit`
- `goodbit`

Definition at line 416 of file `ios_base.h`.



#### 4.582.2.4 openmode

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.

`_Ios_Openmode` is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type `openmode` are:

- `app`
- `ate`
- `binary`
- `in`
- `out`
- `trunc`

Definition at line 447 of file `ios_base.h`.

#### 4.582.2.5 seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir
```

This is an enumerated type.

`_Ios_Seekdir` is implementation-defined. Defined values of type `seekdir` are:

- `beg`
- `cur`, equivalent to `SEEK_CUR` in the C standard library.
- `end`, equivalent to `SEEK_END` in the C standard library.

Definition at line 479 of file `ios_base.h`.

### 4.582.3 Member Enumeration Documentation

#### 4.582.3.1 event

```
enum std::ios_base::event
```

The set of events that may be passed to an event callback.

`erase_event` is used during `~ios()` and `copyfmt()`. `imbue_event` is used during `imbue()`. `copyfmt_event` is used during `copyfmt()`.

Definition at line 512 of file `ios_base.h`.

#### 4.582.4 Constructor & Destructor Documentation

##### 4.582.4.1 ~ios\_base()

```
virtual std::ios_base::~~ios_base () [virtual]
```

Invokes each callback with `erase_event`. Destroys local storage.

Note that the `ios_base` object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with `erase_event` (unless `copyfmt` is used).

#### 4.582.5 Member Function Documentation

##### 4.582.5.1 \_M\_getloc()

```
const locale& std::ios_base::_M_getloc () const [inline]
```

Locale access.

Returns

A reference to the current locale.

Like `getloc` above, but returns a reference instead of generating a copy.

Definition at line 804 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _Inlter >::do_get()`, `std::money_get< _CharT, _Inlter >::do_get()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::time_get< _CharT, _Inlter >::do_get_date()`, `std::time_get< _CharT, _Inlter >::do_get_monthname()`, `std::time_get< _CharT, _Inlter >::do_get_time()`, `std::time_get< _CharT, _Inlter >::do_get_weekday()`, `std::time_put< _CharT, _Outlter >::do_put()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::time_get< _CharT, _Inlter >::get()`, and `std::time_put< _CharT, _Outlter >::put()`.

##### 4.582.5.2 flags() [1/2]

```
fmtflags std::ios_base::flags () const [inline]
```

Access to format flags.

Returns

The format control flags for both input and output.

Definition at line 649 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, `std::num_get< _CharT, _Inlter >::do_get()`, `std::num_put< _CharT, _Outlter >::do_put()`, `std::basic_ostream< char >::operator<<()`, `std::operator<<()`, `std::__detail::operator>>()`, and `std::operator>>()`.

##### 4.582.5.3 flags() [2/2]

```
fmtflags std::ios_base::flags (
 fmtflags __fmtfl) [inline]
```

Setting new format flags all at once.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__fmtfl</code> | The new flags to set. |
|----------------------|-----------------------|

## Returns

The previous format control flags.

This function overwrites all the format flags with `__fmtfl`.

Definition at line 660 of file `ios_base.h`.

## 4.582.5.4 getloc()

```
locale std::ios_base::getloc () const [inline]
```

Locale access.

## Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Definition at line 793 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::copyfmt()`, `std::money_put<_CharT, _OutIter>::do_put()`, and `std::ws()`.

## 4.582.5.5 imbue()

```
locale std::ios_base::imbue (
 const locale & __loc) throw ()
```

Setting a new locale.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with `imbue_event`.

Referenced by `std::basic_ios< char, _Traits >::imbue()`.

#### 4.582.5.6 `iword()`

```
long& std::ios_base::iword (
 int __ix) [inline]
```

Access to integer array.

##### Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

##### Returns

A reference to an integer associated with the index.

The `iword` function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.

The implementation reserves several indices. You should use `xalloc` to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 839 of file `ios_base.h`.

#### 4.582.5.7 `precision()` [1/2]

```
streamsize std::ios_base::precision () const [inline]
```

Flags access.

##### Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

Definition at line 719 of file `ios_base.h`.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::operator<<()`.

#### 4.582.5.8 `precision()` [2/2]

```
streamsize std::ios_base::precision (
 streamsize __prec) [inline]
```

Changing flags.

## Parameters

|                     |                          |
|---------------------|--------------------------|
| <code>__prec</code> | The new precision value. |
|---------------------|--------------------------|

## Returns

The previous value of precision().

Definition at line 728 of file ios\_base.h.

## 4.582.5.9 pword()

```
void*& std::ios_base::pword (
 int __ix) [inline]
```

Access to void pointer array.

## Parameters

|                   |                       |
|-------------------|-----------------------|
| <code>__ix</code> | Index into the array. |
|-------------------|-----------------------|

## Returns

A reference to a void\* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.

The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

Definition at line 860 of file ios\_base.h.

## 4.582.5.10 register\_callback()

```
void std::ios_base::register_callback (
 event_callback __fn,
 int __index)
```

Add the callback \_\_fn with parameter \_\_index.

**Parameters**

|                      |                                                   |
|----------------------|---------------------------------------------------|
| <code>__fn</code>    | The function to add.                              |
| <code>__index</code> | The integer to pass to the function when invoked. |

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

**4.582.5.11** `setf()` [1/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl) [inline]
```

Setting new format flags.

**Parameters**

|                      |                          |
|----------------------|--------------------------|
| <code>__fmtfl</code> | Additional flags to set. |
|----------------------|--------------------------|

**Returns**

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

Definition at line 676 of file `ios_base.h`.

Referenced by `std::__detail::operator>>()`.

**4.582.5.12** `setf()` [2/2]

```
fmtflags std::ios_base::setf (
 fmtflags __fmtfl,
 fmtflags __mask) [inline]
```

Setting new format flags.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__fmtfl</code> | Additional flags to set.          |
| <code>__mask</code>  | The flags mask for <i>fmtfl</i> . |

**Returns**

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

Definition at line 693 of file `ios_base.h`.

#### 4.582.5.13 sync\_with\_stdio()

```
static bool std::ios_base::sync_with_stdio (
 bool __sync = true) [static]
```

Interaction with the standard C I/O objects.

##### Parameters

|                     |                                |
|---------------------|--------------------------------|
| <code>__sync</code> | Whether to synchronize or not. |
|---------------------|--------------------------------|

##### Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., `stdout`) and the standard C++ objects (e.g., `cout`). User-declared streams are unaffected. See <https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary>

#### 4.582.5.14 unsetf()

```
void std::ios_base::unsetf (
 fmtflags __mask) [inline]
```

Clearing format flags.

##### Parameters

|                     |                     |
|---------------------|---------------------|
| <code>__mask</code> | The flags to unset. |
|---------------------|---------------------|

This function clears `__mask` in the format flags.

Definition at line 708 of file `ios_base.h`.

#### 4.582.5.15 width() [1/2]

```
streamsize std::ios_base::width () const [inline]
```

Flags access.

**Returns**

The minimum field width to generate on output operations.

*Minimum field width* refers to the number of characters.

Definition at line 742 of file ios\_base.h.

Referenced by `std::basic_ios< char, _Traits >::copyfmt()`, and `std::num_put< _CharT, _OutIter >::do_put()`.

**4.582.5.16 width()** [2/2]

```
streamsize std::ios_base::width (
 streamsize __wide) [inline]
```

Changing flags.

**Parameters**

|                     |                      |
|---------------------|----------------------|
| <code>__wide</code> | The new width value. |
|---------------------|----------------------|

**Returns**

The previous value of `width()`.

Definition at line 751 of file ios\_base.h.

**4.582.5.17 xalloc()**

```
static int std::ios_base::xalloc () throw () [static]
```

Access to unique indices.

**Returns**

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the `iword` and `pword` functions. The expectation is that an application calls `xalloc` in order to obtain an index in the `iword` and `pword` arrays that can be used without fear of conflict.

The implementation maintains a static variable that is incremented and returned on each invocation. `xalloc` is guaranteed to return an index that is safe to use in the `iword` and `pword` arrays.



#### 4.582.6 Member Data Documentation

##### 4.582.6.1 adjustfield

```
const fmtflags std::ios_base::adjustfield [static]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.

Definition at line 396 of file `ios_base.h`.

Referenced by `std::num_put<_CharT, _OutIter>::do_put()`, `std::internal()`, `std::left()`, and `std::right()`.

##### 4.582.6.2 app

```
const openmode std::ios_base::app [static]
```

Seek to end before each write.

Definition at line 450 of file `ios_base.h`.

##### 4.582.6.3 ate

```
const openmode std::ios_base::ate [static]
```

Open and seek to end immediately after opening.

Definition at line 453 of file `ios_base.h`.

##### 4.582.6.4 badbit

```
const iostate std::ios_base::badbit [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

Definition at line 420 of file `ios_base.h`.

Referenced by `std::basic_ios<char, _Traits>::bad()`, `std::basic_ios<char, _Traits>::fail()`, and `std::operator<<()`.

#### 4.582.6.5 basefield

```
const fmtflags std::ios_base::basefield [static]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

Definition at line 399 of file `ios_base.h`.

Referenced by `std::dec()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_put()`, `std::hex()`, `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

#### 4.582.6.6 beg

```
const seekdir std::ios_base::beg [static]
```

Request a seek relative to the beginning of the stream.

Definition at line 482 of file `ios_base.h`.

#### 4.582.6.7 binary

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↵filestreams.binary](https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary).

Definition at line 458 of file `ios_base.h`.

#### 4.582.6.8 boolalpha

```
const fmtflags std::ios_base::boolalpha [static]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Definition at line 344 of file `ios_base.h`.

Referenced by `std::boolalpha()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::num_put< _CharT, _OutIter >::do_↵put()`, and `std::noboolalpha()`.

#### 4.582.6.9 cur

```
const seekdir std::ios_base::cur [static]
```

Request a seek relative to the current position within the sequence.

Definition at line 485 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.582.6.10 dec

```
const fmtflags std::ios_base::dec [static]
```

Converts integer input or generates integer output in decimal base.

Definition at line 347 of file ios\_base.h.

Referenced by std::dec().

#### 4.582.6.11 end

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

Definition at line 488 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff().

#### 4.582.6.12 eofbit

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

Definition at line 423 of file ios\_base.h.

Referenced by std::time\_get< \_CharT, \_InIter >::do\_get(), std::num\_get< \_CharT, \_InIter >::do\_get(), std::time\_get< \_CharT, \_InIter >::do\_get\_date(), std::time\_get< \_CharT, \_InIter >::do\_get\_monthname(), std::time\_get< \_CharT, \_InIter >::do\_get\_time(), std::time\_get< \_CharT, \_InIter >::do\_get\_weekday(), std::time\_get< \_CharT, \_InIter >::do\_get\_year(), std::basic\_ios< char, \_Traits >::eof(), std::time\_get< \_CharT, \_InIter >::get(), std::basic\_istream< char >::putback(), std::basic\_istream< char >::seekg(), std::basic\_istream< char >::unget(), and std::ws().

#### 4.582.6.13 failbit

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Definition at line 428 of file `ios_base.h`.

Referenced by `std::num_get< _CharT, _InIter >::do_get()`, `std::time_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ios< char, _Traits >::fail()`, `std::time_get< _CharT, _InIter >::get()`, and `std::basic_ostream< _CharT, _Traits >::sentry↵::sentry()`.

#### 4.582.6.14 fixed

```
const fmtflags std::ios_base::fixed [static]
```

Generate floating-point output in fixed-point notation.

Definition at line 350 of file `ios_base.h`.

Referenced by `std::fixed()`, and `std::hexfloat()`.

#### 4.582.6.15 floatfield

```
const fmtflags std::ios_base::floatfield [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Definition at line 402 of file `ios_base.h`.

Referenced by `std::defaultfloat()`, `std::fixed()`, `std::hexfloat()`, and `std::scientific()`.

#### 4.582.6.16 goodbit

```
const iostate std::ios_base::goodbit [static]
```

Indicates all is well.

Definition at line 431 of file `ios_base.h`.

Referenced by `std::time_get< _CharT, _InIter >::do_get()`, `std::num_get< _CharT, _InIter >::do_get()`, `std::time↵_get< _CharT, _InIter >::do_get_monthname()`, `std::time_get< _CharT, _InIter >::do_get_weekday()`, `std::time_get< _CharT, _InIter >::do_get_year()`, `std::basic_ostream< char >::flush()`, `std::basic_istream< char >::get()`, `std::time↵_get< _CharT, _InIter >::get()`, `std::basic_istream< char >::getline()`, `std::basic_istream< char >::ignore()`, `std::basic↵_ostream< char >::operator<<()`, `std::basic_istream< char >::operator>>()`, `std::operator>>()`, `std::basic_istream< char >::peek()`, `std::basic_ostream< char >::put()`, `std::basic_istream< char >::putback()`, `std::basic_istream< char >::read()`, `std::basic_istream< char >::readsom↵e()`, `std::basic_istream< char >::seekg()`, `std::basic_ostream< char >::seekp()`, `std::basic_istream< _CharT, _Traits >::sentry::sentry()`, `std::basic_istream< char >::sync()`, and `std↵::basic_istream< char >::unget()`.

#### 4.582.6.17 hex

```
const fmtflags std::ios_base::hex [static]
```

Converts integer input or generates integer output in hexadecimal base.

Definition at line 353 of file ios\_base.h.

Referenced by std::num\_get< \_CharT, \_InIter >::do\_get(), std::num\_put< \_CharT, \_OutIter >::do\_put(), std::hex(), and std::basic\_ostream< char >::operator<<().

#### 4.582.6.18 in

```
const openmode std::ios_base::in [static]
```

Open for input. Default for ifstream and fstream.

Definition at line 461 of file ios\_base.h.

Referenced by std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::overflow(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekoff(), std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::seekpos(), and std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >::underflow().

#### 4.582.6.19 internal

```
const fmtflags std::ios_base::internal [static]
```

Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.

Definition at line 358 of file ios\_base.h.

Referenced by std::internal().

#### 4.582.6.20 left

```
const fmtflags std::ios_base::left [static]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Definition at line 362 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), and std::left().

**4.582.6.21 oct**

```
const fmtflags std::ios_base::oct [static]
```

Converts integer input or generates integer output in octal base.

Definition at line 365 of file `ios_base.h`.

Referenced by `std::oct()`, and `std::basic_ostream< char >::operator<<()`.

**4.582.6.22 out**

```
const openmode std::ios_base::out [static]
```

Open for output. Default for `ofstream` and `fstream`.

Definition at line 464 of file `ios_base.h`.

Referenced by `std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail()`, `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff()`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos()`.

**4.582.6.23 right**

```
const fmtflags std::ios_base::right [static]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

Definition at line 369 of file `ios_base.h`.

Referenced by `std::right()`.

**4.582.6.24 scientific**

```
const fmtflags std::ios_base::scientific [static]
```

Generates floating-point output in scientific notation.

Definition at line 372 of file `ios_base.h`.

Referenced by `std::hexfloat()`, and `std::scientific()`.

#### 4.582.6.25 showbase

```
const fmtflags std::ios_base::showbase [static]
```

Generates a prefix indicating the numeric base of generated integer output.

Definition at line 376 of file ios\_base.h.

Referenced by std::num\_put< \_CharT, \_OutIter >::do\_put(), std::noshowbase(), and std::showbase().

#### 4.582.6.26 showpoint

```
const fmtflags std::ios_base::showpoint [static]
```

Generates a decimal-point character unconditionally in generated floating-point output.

Definition at line 380 of file ios\_base.h.

Referenced by std::noshowpoint(), and std::showpoint().

#### 4.582.6.27 showpos

```
const fmtflags std::ios_base::showpos [static]
```

Generates a + sign in non-negative generated numeric output.

Definition at line 383 of file ios\_base.h.

Referenced by std::noshowpos(), and std::showpos().

#### 4.582.6.28 skipws

```
const fmtflags std::ios_base::skipws [static]
```

Skips leading white space before certain input operations.

Definition at line 386 of file ios\_base.h.

Referenced by std::noskipws(), and std::skipws().

#### 4.582.6.29 trunc

```
const openmode std::ios_base::trunc [static]
```

Truncate an existing stream when opening. Default for `ofstream`.

Definition at line 467 of file `ios_base.h`.

#### 4.582.6.30 unitbuf

```
const fmtflags std::ios_base::unitbuf [static]
```

Flushes output after each output operation.

Definition at line 389 of file `ios_base.h`.

Referenced by `std::nounitbuf()`, and `std::unitbuf()`.

#### 4.582.6.31 uppercase

```
const fmtflags std::ios_base::uppercase [static]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.

Definition at line 393 of file `ios_base.h`.

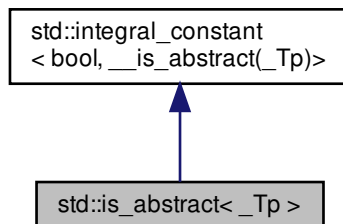
Referenced by `std::num_put<_CharT, _OutIter >::do_put()`, `std::nouppercase()`, and `std::uppercase()`.

The documentation for this class was generated from the following file:

- [ios\\_base.h](#)

### 4.583 std::is\_abstract<\_Tp> Struct Template Reference

Inheritance diagram for `std::is_abstract<_Tp>`:





## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.583.1 Detailed Description

```
template<typename _Tp>
struct std::is_abstract< _Tp >
```

`is_abstract`

Definition at line 736 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.584 `std::is_arithmetic< _Tp >` Struct Template Reference

Inherits type< `is_integral< _Tp >`, `is_floating_point< _Tp >` >.

## 4.584.1 Detailed Description

```
template<typename _Tp>
struct std::is_arithmetic< _Tp >
```

`is_arithmetic`

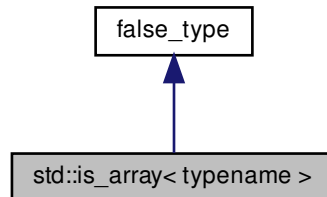
Definition at line 534 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.585 `std::is_array< typename >` Struct Template Reference

Inheritance diagram for `std::is_array< typename >`:



##### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr `_Tp` **value**

##### 4.585.1 Detailed Description

```
template<typename>
struct std::is_array< typename >
```

`is_array`

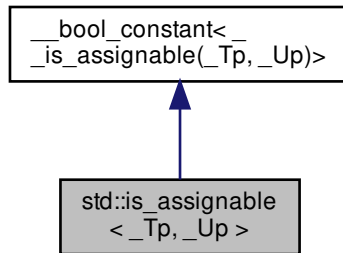
Definition at line 399 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.586 std::is\_assignable&lt; \_Tp, \_Up &gt; Struct Template Reference

Inheritance diagram for std::is\_assignable< \_Tp, \_Up >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.586.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_assignable< _Tp, _Up >
```

is\_assignable

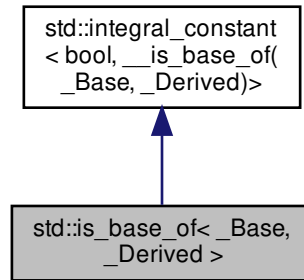
Definition at line 1069 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.587 std::is\_base\_of<\_Base, \_Derived> Struct Template Reference

Inheritance diagram for std::is\_base\_of<\_Base, \_Derived>:



##### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr bool **value**

##### 4.587.1 Detailed Description

```
template<typename _Base, typename _Derived>
struct std::is_base_of<_Base, _Derived>
```

is\_base\_of

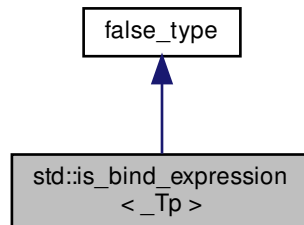
Definition at line 1411 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.588 std::is\_bind\_expression&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_bind\_expression< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.588.1 Detailed Description

```
template<typename _Tp>
struct std::is_bind_expression< _Tp >
```

Determines if the given type \_Tp is a function object that should be treated as a subexpression when evaluating calls to function objects returned by bind().

C++11 [func.bind.isbind].

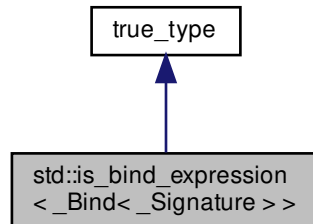
Definition at line 184 of file functional.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.589 `std::is_bind_expression<_Bind<_Signature>>` Struct Template Reference

Inheritance diagram for `std::is_bind_expression<_Bind<_Signature>>`:



##### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.589.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression<_Bind<_Signature>>
```

Class template `_Bind` is always a bind expression.

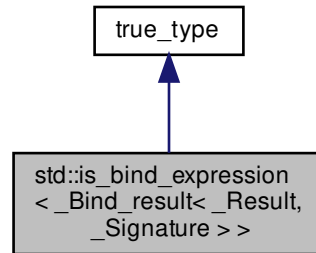
Definition at line 670 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.590 `std::is_bind_expression< _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.590.1 Detailed Description

```

template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

```

Class template `_Bind_result` is always a bind expression.

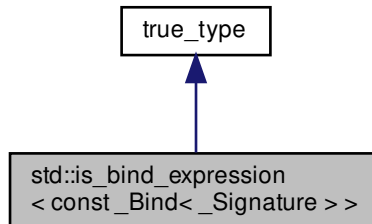
Definition at line 702 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.591 `std::is_bind_expression< const _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const _Bind< _Signature > >`:



##### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.591.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

Definition at line 678 of file `functional`.

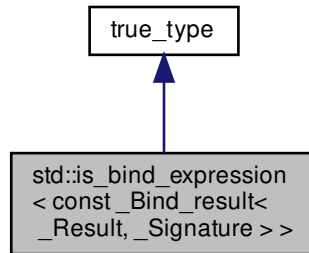
The documentation for this struct was generated from the following file:

- [functional](#)



4.592 `std::is_bind_expression< const _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.592.1 Detailed Description

```

template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

```

Class template `_Bind_result` is always a bind expression.

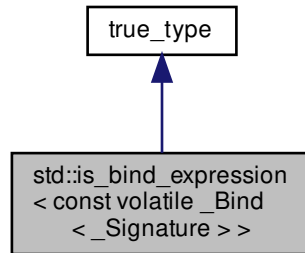
Definition at line 710 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.593 `std::is_bind_expression< const volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind< _Signature > >`:



##### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.593.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< const volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

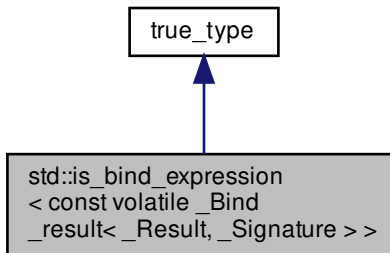
Definition at line 694 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.594 `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.594.1 Detailed Description

```
template<typename _Result, typename _Signature>
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
```

Class template `_Bind_result` is always a bind expression.

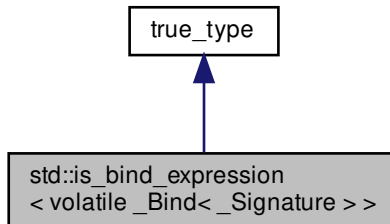
Definition at line 726 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.595 `std::is_bind_expression< volatile _Bind< _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< volatile _Bind< _Signature > >`:



##### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.595.1 Detailed Description

```
template<typename _Signature>
struct std::is_bind_expression< volatile _Bind< _Signature > >
```

Class template `_Bind` is always a bind expression.

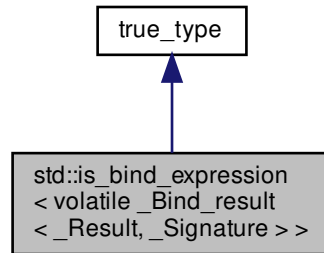
Definition at line 686 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.596 `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >` Struct Template Reference

Inheritance diagram for `std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >`:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.596.1 Detailed Description

```

template<typename _Result, typename _Signature>
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

```

Class template `_Bind_result` is always a bind expression.

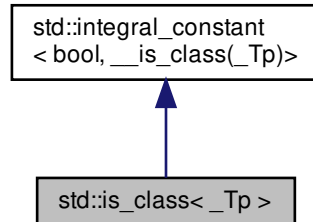
Definition at line 718 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.597 `std::is_class<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_class<_Tp>`:



##### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr bool **value**

#### 4.597.1 Detailed Description

```
template<typename _Tp>
struct std::is_class<_Tp>
```

`is_class`

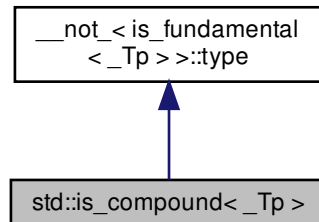
Definition at line 484 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.598 std::is\_compound< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_compound< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.598.1 Detailed Description

```
template<typename _Tp>
struct std::is_compound< _Tp >
```

is\_compound

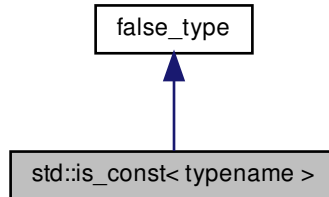
Definition at line 564 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.599 `std::is_const< typename >` Struct Template Reference

Inheritance diagram for `std::is_const< typename >`:



##### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr `_Tp` **value**

##### 4.599.1 Detailed Description

```
template<typename>
struct std::is_const< typename >
```

`is_const`

Definition at line 234 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 4.600 `std::is_constructible< _Tp, _Args >` Struct Template Reference

Inherits `std::__is_constructible_impl< _Tp, _Args >`.

### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.600.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_constructible< _Tp, _Args >
```

`is_constructible`

Definition at line 906 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.601 `std::is_convertible< _From, _To >` Struct Template Reference

Inherits `type< _From, _To >`.

#### 4.601.1 Detailed Description

```
template<typename _From, typename _To>
struct std::is_convertible< _From, _To >
```

`is_convertible`

Definition at line 1447 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.602 `std::is_copy_assignable<_Tp>` Struct Template Reference

Inherits `type<_Tp>`.

##### 4.602.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_assignable<_Tp>
```

`is_copy_assignable`

Definition at line 1090 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.603 `std::is_copy_constructible<_Tp>` Struct Template Reference

Inherits `std::__is_copy_constructible_impl<_Tp, bool>`.

Inherited by `std::__is_copy_insertable< allocator<_Tp>>`.

##### 4.603.1 Detailed Description

```
template<typename _Tp>
struct std::is_copy_constructible<_Tp>
```

`is_copy_constructible`

Definition at line 936 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.604 `std::is_default_constructible<_Tp>` Struct Template Reference

Inherits `type<_Tp>`.

## 4.604.1 Detailed Description

```
template<typename _Tp>
struct std::is_default_constructible< _Tp >
```

is\_default\_constructible

Definition at line 915 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.605 std::is\_destructible&lt; \_Tp &gt; Struct Template Reference

Inherits type< \_Tp >.

## 4.605.1 Detailed Description

```
template<typename _Tp>
struct std::is_destructible< _Tp >
```

is\_destructible

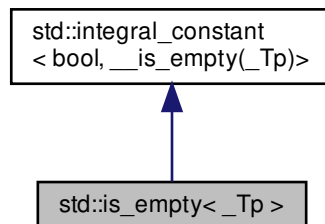
Definition at line 841 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.606 std::is\_empty&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_empty< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 4.606.1 Detailed Description

```
template<typename _Tp>
struct std::is_empty< _Tp >
```

is\_empty

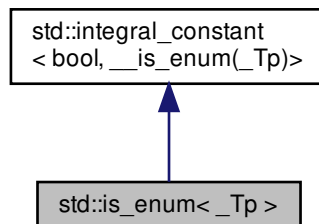
Definition at line 715 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.607 std::is\_enum< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_enum< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.607.1 Detailed Description

```
template<typename _Tp>
struct std::is_enum< _Tp >
```

is\_enum

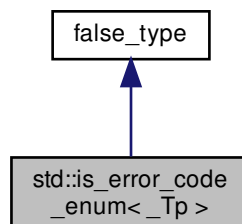
Definition at line 472 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.608 std::is\_error\_code\_enum&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_error\_code\_enum< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.608.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_code_enum< _Tp >
```

is\_error\_code\_enum

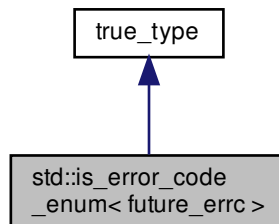
Definition at line 60 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 4.609 std::is\_error\_code\_enum< future\_errc > Struct Template Reference

Inheritance diagram for std::is\_error\_code\_enum< future\_errc >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.609.1 Detailed Description

```
template<>
struct std::is_error_code_enum< future_errc >
```

Specialization.

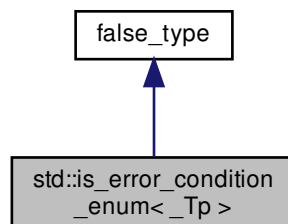
Definition at line 76 of file future.

The documentation for this struct was generated from the following file:

- [future](#)

4.610 `std::is_error_condition_enum< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_error_condition_enum< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.610.1 Detailed Description

```
template<typename _Tp>
struct std::is_error_condition_enum< _Tp >
```

is\_error\_condition\_enum

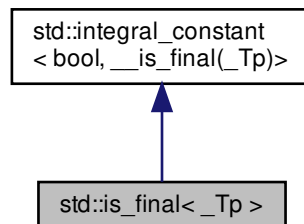
Definition at line 64 of file system\_error.

The documentation for this struct was generated from the following file:

- [system\\_error](#)

#### 4.611 std::is\_final< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_final< \_Tp >:





## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.611.1 Detailed Description

```
template<typename _Tp>
struct std::is_final< _Tp >
```

is\_final

Definition at line 729 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.612 `std::is_floating_point< _Tp >` Struct Template Reference

Inherits `type< __remove_cv_t< _Tp > >`.

## 4.612.1 Detailed Description

```
template<typename _Tp>
struct std::is_floating_point< _Tp >
```

is\_floating\_point

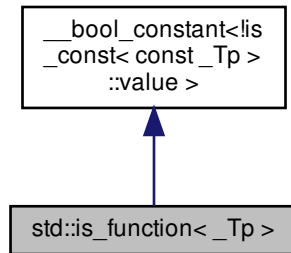
Definition at line 393 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.613 `std::is_function<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_function<_Tp>`:



##### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.613.1 Detailed Description

```
template<typename _Tp>
struct std::is_function<_Tp>
```

`is_function`

Definition at line 191 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.614 std::is\_fundamental< \_Tp > Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, is\_void< \_Tp >, is\_null\_pointer< \_Tp > >.

### 4.614.1 Detailed Description

```
template<typename _Tp>
struct std::is_fundamental< _Tp >
```

is\_fundamental

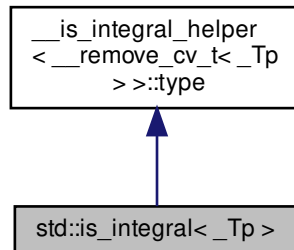
Definition at line 540 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.615 std::is\_integral< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_integral< \_Tp >:



### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 4.615.1 Detailed Description

```
template<typename _Tp>
struct std::is_integral<_Tp>
```

`is_integral`

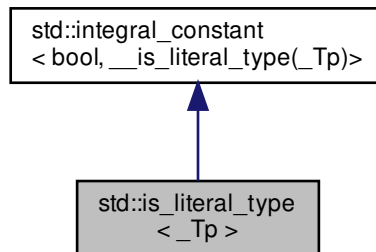
Definition at line 365 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.616 `std::is_literal_type<_Tp>` Struct Template Reference

Inheritance diagram for `std::is_literal_type<_Tp>`:



#### Public Types

- typedef [integral\\_constant](#)< bool, `__v` > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.616.1 Detailed Description

```
template<typename _Tp>
struct std::is_literal_type< _Tp >
```

is\_literal\_type

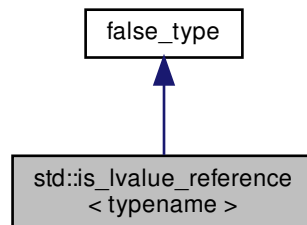
Definition at line 706 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.617 std::is\_lvalue\_reference&lt; typename &gt; Struct Template Reference

Inheritance diagram for std::is\_lvalue\_reference< typename >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr `_Tp` **value**

#### 4.617.1 Detailed Description

```
template<typename>
struct std::is_lvalue_reference< typename >
```

`is_lvalue_reference`

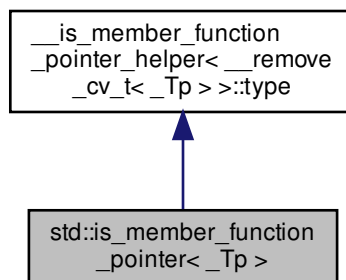
Definition at line 426 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.618 `std::is_member_function_pointer< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_member_function_pointer< _Tp >`:



#### Public Types

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr `_Tp` **value**

## 4.618.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_function_pointer< _Tp >
```

is\_member\_function\_pointer

Definition at line 466 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.619 std::is\_member\_object\_pointer&lt; \_Tp &gt; Struct Template Reference

Inherits type< \_\_remove\_cv\_t< \_Tp > >.

## 4.619.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_object_pointer< _Tp >
```

is\_member\_object\_pointer

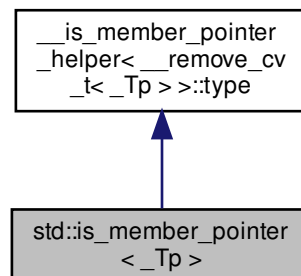
Definition at line 452 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.620 std::is\_member\_pointer&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_member\_pointer< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.620.1 Detailed Description

```
template<typename _Tp>
struct std::is_member_pointer< _Tp >
```

is\_member\_pointer

Definition at line 553 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.621 std::is\_move\_assignable< \_Tp > Struct Template Reference

Inherits type< \_Tp >.

#### 4.621.1 Detailed Description

```
template<typename _Tp>
struct std::is_move_assignable< _Tp >
```

is\_move\_assignable

Definition at line 1111 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)



## 4.622 `std::is_move_constructible< _Tp >` Struct Template Reference

Inherits `std::__is_move_constructible_impl< _Tp, bool >`.

Inherited by `std::__is_move_insertable< allocator< _Tp > >`.

### 4.622.1 Detailed Description

```
template<typename _Tp>
struct std::is_move_constructible< _Tp >
```

`is_move_constructible`

Definition at line 957 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.623 `std::is_nothrow_assignable< _Tp, _Up >` Struct Template Reference

Inherits `std::__is_nothrow_assignable_impl< _Tp, _Up >`.

### 4.623.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >
```

`is_nothrow_assignable`

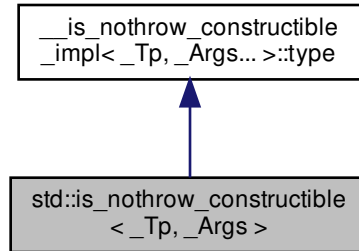
Definition at line 1131 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.624 `std::is_nothrow_constructible<_Tp, _Args>` Struct Template Reference

Inheritance diagram for `std::is_nothrow_constructible<_Tp, _Args>`:



##### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.624.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible<_Tp, _Args>
```

`is_nothrow_constructible`

Definition at line 1008 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.625 std::is\_nothrow\_copy\_assignable< \_Tp > Struct Template Reference

Inherits std::\_\_is\_nt\_copy\_assignable\_impl< \_Tp, bool >.

### 4.625.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_assignable< _Tp >
```

is\_nothrow\_copy\_assignable

Definition at line 1152 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.626 std::is\_nothrow\_copy\_constructible< \_Tp > Struct Template Reference

Inherits type< \_Tp >.

### 4.626.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_copy_constructible< _Tp >
```

is\_nothrow\_copy\_constructible

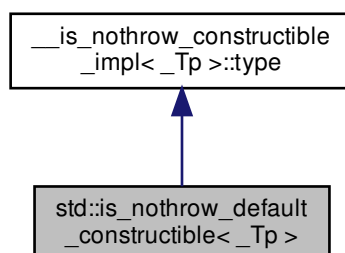
Definition at line 1039 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.627 std::is\_nothrow\_default\_constructible< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_nothrow\_default\_constructible< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

#### 4.627.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_default_constructible< _Tp >
```

is\_nothrow\_default\_constructible

Definition at line 1017 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.628 std::is\_nothrow\_destructible< \_Tp > Struct Template Reference

Inherits type< \_Tp >.

#### 4.628.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_destructible< _Tp >
```

is\_nothrow\_destructible

Definition at line 892 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.629 `std::is_nothrow_move_assignable< _Tp >` Struct Template Reference

Inherits `std::__is_nt_move_assignable_impl< _Tp, bool >`.

### 4.629.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_assignable< _Tp >
```

`is_nothrow_move_assignable`

Definition at line 1173 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.630 `std::is_nothrow_move_constructible< _Tp >` Struct Template Reference

Inherits `type< _Tp >`.

### 4.630.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_move_constructible< _Tp >
```

`is_nothrow_move_constructible`

Definition at line 1060 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.631 `std::is_nothrow_swappable< _Tp >` Struct Template Reference

Inherits `type< _Tp >`.

#### 4.631.1 Detailed Description

```
template<typename _Tp>
struct std::is_nothrow_swappable< _Tp >
```

is\_nothrow\_swappable

Definition at line 2729 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.632 std::is\_nothrow\_swappable\_with< \_Tp, \_Up > Struct Template Reference

Inherits type< \_Tp, \_Up >.

##### 4.632.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_nothrow_swappable_with< _Tp, _Up >
```

is\_nothrow\_swappable\_with

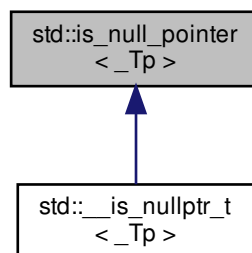
Definition at line 2816 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.633 std::is\_null\_pointer< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_null\_pointer< \_Tp >:



## 4.633.1 Detailed Description

```
template<typename _Tp>
struct std::is_null_pointer< _Tp >
```

is\_null\_pointer (LWG 2247).

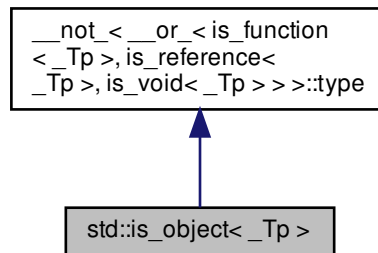
Definition at line 513 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.634 std::is\_object&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_object< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

#### 4.634.1 Detailed Description

```
template<typename _Tp>
struct std::is_object< _Tp >
```

is\_object

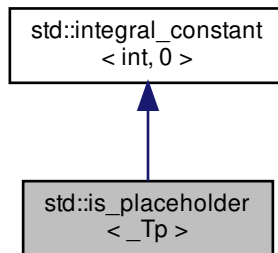
Definition at line 547 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.635 std::is\_placeholder< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_placeholder< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< int, \_\_v > **type**
- typedef int **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr int **value**



## 4.635.1 Detailed Description

```
template<typename _Tp>
struct std::is_placeholder<_Tp>
```

Determines if the given type `_Tp` is a placeholder in a `bind()` expression and, if so, which placeholder it is.

C++11 [func.bind.isplace].

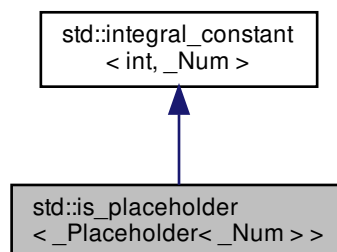
Definition at line 195 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

4.636 `std::is_placeholder<_Placeholder<_Num>>` Struct Template Reference

Inheritance diagram for `std::is_placeholder<_Placeholder<_Num>>`:



## Public Types

- typedef `integral_constant<int, __v>` **type**
- typedef `int` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr `value_type` **operator()** () const noexcept

### Static Public Attributes

- static constexpr int **value**

#### 4.636.1 Detailed Description

```
template<int _Num>
struct std::is_placeholder< _Placeholder< _Num > >
```

Partial specialization of `is_placeholder` that provides the placeholder number for the placeholder objects defined by `libstdc++`.

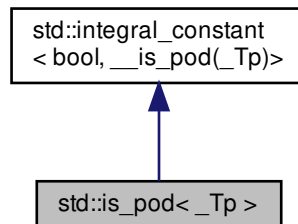
Definition at line 258 of file `functional`.

The documentation for this struct was generated from the following file:

- [functional](#)

#### 4.637 `std::is_pod< _Tp >` Struct Template Reference

Inheritance diagram for `std::is_pod< _Tp >`:



### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.637.1 Detailed Description

```
template<typename _Tp>
struct std::is_pod< _Tp >
```

is\_pod (deprecated in C++20)

Definition at line 695 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.638 std::is\_pointer&lt; \_Tp &gt; Struct Template Reference

Inherits type< \_\_remove\_cv\_t< \_Tp > >.

## 4.638.1 Detailed Description

```
template<typename _Tp>
struct std::is_pointer< _Tp >
```

is\_pointer

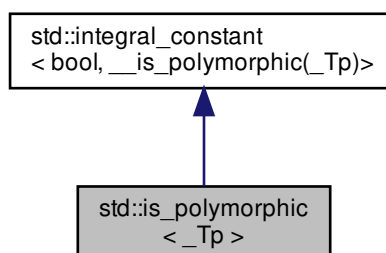
Definition at line 420 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.639 std::is\_polymorphic&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_polymorphic< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 4.639.1 Detailed Description

```
template<typename _Tp>
struct std::is_polymorphic< _Tp >
```

is\_polymorphic

Definition at line 721 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.640 std::is\_reference< \_Tp > Struct Template Reference

Inherits type< is\_lvalue\_reference< \_Tp >, is\_rvalue\_reference< \_Tp > >.

#### 4.640.1 Detailed Description

```
template<typename _Tp>
struct std::is_reference< _Tp >
```

is\_reference

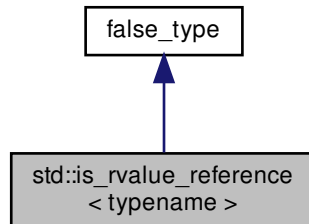
Definition at line 189 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.641 `std::is_rvalue_reference< typename >` Struct Template Reference

Inheritance diagram for `std::is_rvalue_reference< typename >`:

**Public Types**

- typedef [integral\\_constant](#)< `_Tp`, `__v` > **type**
- typedef `_Tp` **value\_type**

**Public Member Functions**

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

**Static Public Attributes**

- static constexpr `_Tp` **value**

## 4.641.1 Detailed Description

```
template<typename>
struct std::is_rvalue_reference< typename >
```

`is_rvalue_reference`

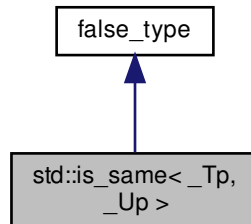
Definition at line 435 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.642 `std::is_same<_Tp, _Up>` Struct Template Reference

Inheritance diagram for `std::is_same<_Tp, _Up>`:



##### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.642.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_same<_Tp, _Up>
```

`is_same`

Definition at line 582 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.643 std::is\_scalar&lt; \_Tp &gt; Struct Template Reference

Inherits type< is\_arithmetic< \_Tp >, is\_enum< \_Tp >, is\_pointer< \_Tp >, is\_member\_pointer< \_Tp >, is\_null\_pointer< \_Tp > >.

## 4.643.1 Detailed Description

```
template<typename _Tp>
struct std::is_scalar< _Tp >
```

is\_scalar

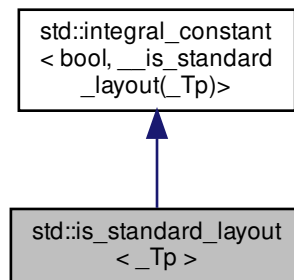
Definition at line 557 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.644 std::is\_standard\_layout&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_standard\_layout< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

##### 4.644.1 Detailed Description

```
template<typename _Tp>
struct std::is_standard_layout< _Tp >
```

is\_standard\_layout

Definition at line 685 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.645 std::is\_swappable< \_Tp > Struct Template Reference

Inherits type< \_Tp >.

##### 4.645.1 Detailed Description

```
template<typename _Tp>
struct std::is_swappable< _Tp >
```

Metafunctions used for detecting swappable types: p0185r1.

is\_swappable

Definition at line 2720 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.646 std::is\_swappable\_with< \_Tp, \_Up > Struct Template Reference

Inherits type< \_Tp, \_Up >.



## 4.646.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_swappable_with< _Tp, _Up >
```

is\_swappable\_with

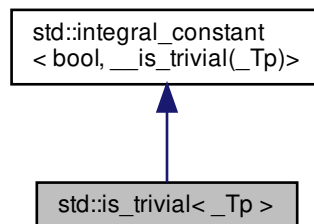
Definition at line 2810 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.647 std::is\_trivial&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_trivial< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

#### 4.647.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivial<_Tp>
```

is\_trivial

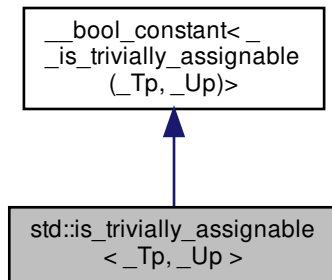
Definition at line 667 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.648 std::is\_trivially\_assignable<\_Tp, \_Up> Struct Template Reference

Inheritance diagram for std::is\_trivially\_assignable<\_Tp, \_Up>:



#### Public Types

- typedef [integral\\_constant](#)<\_Tp, \_\_v> **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**

## 4.648.1 Detailed Description

```
template<typename _Tp, typename _Up>
struct std::is_trivially_assignable< _Tp, _Up >
```

is\_trivially\_assignable

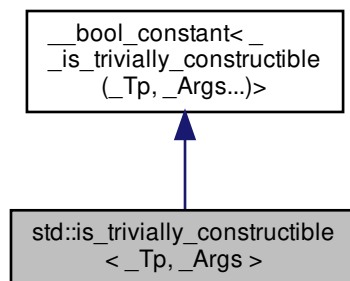
Definition at line 1276 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.649 std::is\_trivially\_constructible&lt; \_Tp, \_Args &gt; Struct Template Reference

Inheritance diagram for std::is\_trivially\_constructible< \_Tp, \_Args >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

#### 4.649.1 Detailed Description

```
template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible< _Tp, _Args >
```

is\_trivially\_constructible

Definition at line 1182 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.650 std::is\_trivially\_copy\_assignable< \_Tp > Struct Template Reference

Inherits std::\_\_is\_trivially\_copy\_assignable\_impl< \_Tp, bool >.

##### 4.650.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_assignable< _Tp >
```

is\_trivially\_copy\_assignable

Definition at line 1297 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.651 std::is\_trivially\_copy\_constructible< \_Tp > Struct Template Reference

Inherits std::\_\_is\_trivially\_copy\_constructible\_impl< \_Tp, bool >.

##### 4.651.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_copy_constructible< _Tp >
```

is\_trivially\_copy\_constructible

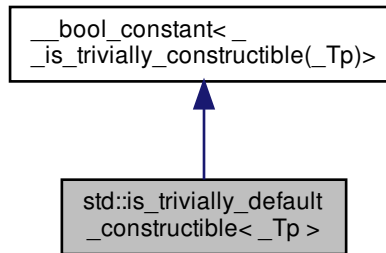
Definition at line 1244 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.652 std::is\_trivially\_default\_constructible&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_trivially\_default\_constructible< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

## 4.652.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_default_constructible< _Tp >
```

is\_trivially\_default\_constructible

Definition at line 1191 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.653 `std::is_trivially_destructible< _Tp >` Struct Template Reference

Inherits `std::__and_<... >`.

##### 4.653.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_destructible< _Tp >
```

`is_trivially_destructible`

Definition at line 1327 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.654 `std::is_trivially_move_assignable< _Tp >` Struct Template Reference

Inherits `std::__is_trivially_move_assignable_impl< _Tp, bool >`.

##### 4.654.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_assignable< _Tp >
```

`is_trivially_move_assignable`

Definition at line 1318 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.655 `std::is_trivially_move_constructible< _Tp >` Struct Template Reference

Inherits `std::__is_trivially_move_constructible_impl< _Tp, bool >`.

## 4.655.1 Detailed Description

```
template<typename _Tp>
struct std::is_trivially_move_constructible< _Tp >
```

is\_trivially\_move\_constructible

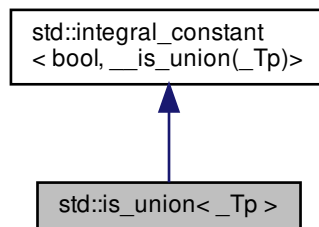
Definition at line 1267 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.656 std::is\_union&lt; \_Tp &gt; Struct Template Reference

Inheritance diagram for std::is\_union< \_Tp >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

#### 4.656.1 Detailed Description

```
template<typename _Tp>
struct std::is_union< _Tp >
```

is\_union

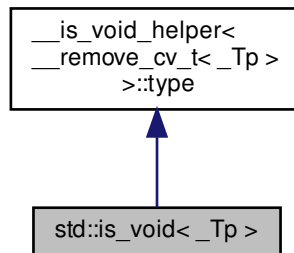
Definition at line 478 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.657 std::is\_void< \_Tp > Struct Template Reference

Inheritance diagram for std::is\_void< \_Tp >:



#### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr \_Tp **value**



## 4.657.1 Detailed Description

```
template<typename _Tp>
struct std::is_void< _Tp >
```

is\_void

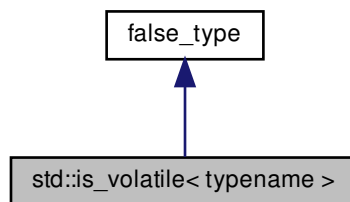
Definition at line 193 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.658 std::is\_volatile&lt; typename &gt; Struct Template Reference

Inheritance diagram for std::is\_volatile< typename >:



## Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr \_Tp **value**

#### 4.658.1 Detailed Description

```
template<typename>
struct std::is_volatile< typename >
```

is\_volatile

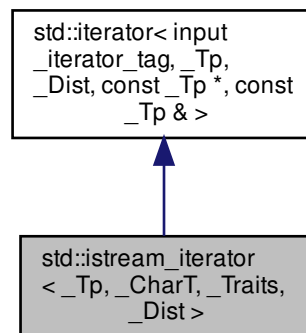
Definition at line 658 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.659 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist > Class Template Reference

Inheritance diagram for std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >:



#### Public Types

- typedef `_CharT` **char\_type**
- typedef `_Dist` [difference\\_type](#)
- typedef [basic\\_istream](#)< `_CharT`, `_Traits` > **istream\_type**
- typedef [input\\_iterator\\_tag](#) `iterator_category`
- typedef `const _Tp *` [pointer](#)
- typedef `const _Tp &` [reference](#)
- typedef `_Traits` **traits\_type**
- typedef `_Tp` [value\\_type](#)

## Public Member Functions

- constexpr `istream_iterator` ()
- `istream_iterator` (`istream_type` &\_\_s)
- `istream_iterator` (const `istream_iterator` &\_\_obj)
- const `_Tp` & `operator*` () const
- `istream_iterator` & `operator++` ()
- `istream_iterator` `operator++` (int)
- const `_Tp` \* `operator->` () const
- `istream_iterator` & `operator=` (const `istream_iterator` &)=default

## Friends

- bool `operator!=` (const `istream_iterator` &\_\_x, const `istream_iterator` &\_\_y)
- bool `operator==` (const `istream_iterator` &\_\_x, const `istream_iterator` &\_\_y)

## 4.659.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
class std::istream_iterator<_Tp, _CharT, _Traits, _Dist>
```

Provides input iterator semantics for streams.

Definition at line 49 of file `stream_iterator.h`.

## 4.659.2 Member Typedef Documentation

4.659.2.1 `difference_type`

```
typedef _Dist std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >↔
::difference_type [inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file `stl_iterator_base_types.h`.

4.659.2.2 `iterator_category`

```
typedef input_iterator_tag std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const ↵
_Tp & >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file `stl_iterator_base_types.h`.

#### 4.659.2.3 pointer

```
typedef const _Tp * std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp &
>::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file `stl_iterator_base_types.h`.

#### 4.659.2.4 reference

```
typedef const _Tp & std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp &
>::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file `stl_iterator_base_types.h`.

#### 4.659.2.5 value\_type

```
typedef _Tp std::iterator< input_iterator_tag , _Tp, _Dist , const _Tp * , const _Tp & >::value_type
[inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

### 4.659.3 Constructor & Destructor Documentation

#### 4.659.3.1 `istream_iterator()` [1/2]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
constexpr std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator () [inline]
```

Construct end of input stream iterator.

Definition at line 67 of file `stream_iterator.h`.

### 4.659.3.2 istream\_iterator() [2/2]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
 istream_type & __s) [inline]
```

Construct start of input stream iterator.

Definition at line 71 of file stream\_iterator.h.

## 4.659.4 Friends And Related Function Documentation

### 4.659.4.1 operator!=

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
bool operator!= (
 const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
 const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [friend]
```

Return true if the iterators refer to different streams, or if one is at end-of-stream and the other is not.

Definition at line 153 of file stream\_iterator.h.

### 4.659.4.2 operator==

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
bool operator== (
 const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
 const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y) [friend]
```

Return true if the iterators refer to the same stream, or are both at end-of-stream.

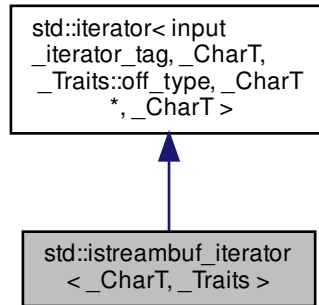
Definition at line 147 of file stream\_iterator.h.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

#### 4.660 `std::istreambuf_iterator<_CharT, _Traits>` Class Template Reference

Inheritance diagram for `std::istreambuf_iterator<_CharT, _Traits>`:



##### Public Types

- typedef `_Traits::off_type` [difference\\_type](#)
  - typedef `input_iterator_tag` [iterator\\_category](#)
  - typedef `_CharT *` [pointer](#)
  - typedef `_CharT` [reference](#)
  - typedef `_CharT` [value\\_type](#)
- 
- typedef `_CharT` [char\\_type](#)
  - typedef `_Traits` [traits\\_type](#)
  - typedef `_Traits::int_type` [int\\_type](#)
  - typedef `basic_istreambuf<_CharT, _Traits>` [streambuf\\_type](#)
  - typedef `basic_istream<_CharT, _Traits>` [istream\\_type](#)

##### Public Member Functions

- constexpr [istreambuf\\_iterator](#) () noexcept
- **istreambuf\_iterator** (const [istreambuf\\_iterator](#) &) noexcept=default
- [istreambuf\\_iterator](#) ([istream\\_type](#) &\_\_s) noexcept
- [istreambuf\\_iterator](#) ([streambuf\\_type](#) \*\_\_s) noexcept
- bool [equal](#) (const [istreambuf\\_iterator](#) &\_\_b) const
- [char\\_type](#) [operator\\*](#) () const
- [istreambuf\\_iterator](#) & [operator++](#) ()
- [istreambuf\\_iterator](#) [operator++](#) (int)
- [istreambuf\\_iterator](#) & **operator=** (const [istreambuf\\_iterator](#) &) noexcept=default

## Friends

- `template<bool _IsMove, typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type __copy_move_a2`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)`
- `template<typename _CharT2, typename _Size >`  
`__enable_if_t< __is_char< _CharT2 >::__value, _CharT2 * > __copy_n_a (istreambuf_iterator< _CharT2 >,`  
`_Size, _CharT2 *)`
- `template<typename _CharT2, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type advance (istreambuf_iterator< _`  
`CharT2 > &, _Distance)`
- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 >::__type copy`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)`
- `template<typename _CharT2 >`  
`__gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 >::__type find`  
`(istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)`

## 4.660.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >
```

Provides input iterator semantics for streambufs.

Definition at line 125 of file iosfwd.

## 4.660.2 Member Typedef Documentation

## 4.660.2.1 char\_type

```
template<typename _CharT , typename _Traits >
typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

Definition at line 66 of file streambuf\_iterator.h.

## 4.660.2.2 difference\_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT
* , _CharT >::difference_type [inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file stl\_iterator\_base\_types.h.

#### 4.660.2.3 int\_type

```
template<typename _CharT , typename _Traits >
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
```

Public typedefs.

Definition at line 68 of file streambuf\_iterator.h.

#### 4.660.2.4 istream\_type

```
template<typename _CharT , typename _Traits >
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
```

Public typedefs.

Definition at line 70 of file streambuf\_iterator.h.

#### 4.660.2.5 iterator\_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _↵
CharT * , _CharT >::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.660.2.6 pointer

```
typedef _CharT * std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _↵
CharT >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file stl\_iterator\_base\_types.h.

#### 4.660.2.7 reference

```
typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT
>::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file stl\_iterator\_base\_types.h.



#### 4.660.2.8 streambuf\_type

```
template<typename _CharT , typename _Traits >
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf_type
```

Public typedefs.

Definition at line 69 of file streambuf\_iterator.h.

#### 4.660.2.9 traits\_type

```
template<typename _CharT , typename _Traits >
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
```

Public typedefs.

Definition at line 67 of file streambuf\_iterator.h.

#### 4.660.2.10 value\_type

```
typedef _CharT std::iterator< input_iterator_tag , _CharT , _Traits::off_type , _CharT * , _CharT
>::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file stl\_iterator\_base\_types.h.

### 4.660.3 Constructor & Destructor Documentation

#### 4.660.3.1 istreambuf\_iterator() [1/3]

```
template<typename _CharT , typename _Traits >
constexpr std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator () [inline], [noexcept]
```

Construct end of input stream iterator.

Definition at line 115 of file streambuf\_iterator.h.

#### 4.660.3.2 `istreambuf_iterator()` [2/3]

```
template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
 istream_type & __s) [inline], [noexcept]
```

Construct start of input stream iterator.

Definition at line 130 of file `streambuf_iterator.h`.

#### 4.660.3.3 `istreambuf_iterator()` [3/3]

```
template<typename _CharT , typename _Traits >
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
 streambuf_type * __s) [inline], [noexcept]
```

Construct start of streambuf iterator.

Definition at line 134 of file `streambuf_iterator.h`.

### 4.660.4 Member Function Documentation

#### 4.660.4.1 `equal()`

```
template<typename _CharT , typename _Traits >
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
 const istreambuf_iterator< _CharT, _Traits > & __b) const [inline]
```

Return true both iterators are end or both are not end.

Definition at line 194 of file `streambuf_iterator.h`.

#### 4.660.4.2 `operator*()`

```
template<typename _CharT , typename _Traits >
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* () const [inline]
```

Return the current character pointed to by iterator. This returns `streambuf.sgetc()`. It cannot be assigned. NB: The result of `operator*()` on an end of stream is undefined.

Definition at line 146 of file `streambuf_iterator.h`.

## 4.660.4.3 operator++() [1/2]

```
template<typename _CharT , typename _Traits >
istreambuf_iterator& std::istreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 162 of file streambuf\_iterator.h.

## 4.660.4.4 operator++() [2/2]

```
template<typename _CharT , typename _Traits >
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
 int) [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

Definition at line 176 of file streambuf\_iterator.h.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [streambuf\\_iterator.h](#)

## 4.661 std::experimental::filesystem::v1::path::iterator Class Reference

## Public Types

- using **difference\_type** = std::ptrdiff\_t
- using **iterator\_category** = [std::bidirectional\\_iterator\\_tag](#)
- using **pointer** = const [path](#) \*
- using **reference** = const [path](#) &
- using **value\_type** = [path](#)

## Public Member Functions

- **iterator** (const [iterator](#) &)=default
- [reference](#) **operator\*** () const
- [iterator](#) & **operator++** ()
- [iterator](#) **operator++** (int)
- [iterator](#) & **operator--** ()
- [iterator](#) **operator--** (int)
- [pointer](#) **operator->** () const
- [iterator](#) & **operator=** (const [iterator](#) &)=default

## Friends

- bool **operator!=** (const [iterator](#) &\_\_lhs, const [iterator](#) &\_\_rhs)
- bool **operator==** (const [iterator](#) &\_\_lhs, const [iterator](#) &\_\_rhs)
- class **path**

### 4.661.1 Detailed Description

An iterator for the components of a path.

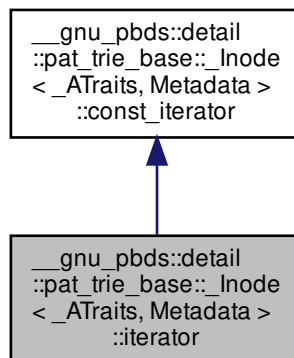
Definition at line 847 of file experimental/bits/fs\_path.h.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

### 4.662 \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata >::iterator Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode<\_ATraits, Metadata >::iterator:



## Public Types

- typedef `_Alloc::difference_type` **difference\_type**
- typedef `std::forward_iterator_tag` **iterator\_category**
- typedef `node_pointer_pointer` **pointer**
- typedef `node_pointer_reference` **reference**
- typedef `node_pointer` **value\_type**

## Public Member Functions

- **iterator** (node\_pointer\_pointer p\_p\_cur=0, node\_pointer\_pointer p\_p\_end=0)
- **bool operator!=** (const [const\\_iterator](#) &other) const
- **bool operator!=** (const [iterator](#) &other) const
- **node\_const\_pointer operator\*** () const
- **node\_pointer operator\*** ()
- [iterator](#) & **operator++** ()
- [iterator](#) **operator++** (int)
- **const node\_pointer\_pointer operator->** () const
- **node\_pointer\_pointer operator->** ()
- **bool operator==** (const [const\\_iterator](#) &other) const
- **bool operator==** (const [iterator](#) &other) const

## Public Attributes

- **node\_pointer\_pointer m\_p\_p\_cur**
- **node\_pointer\_pointer m\_p\_p\_end**

## 4.662.1 Detailed Description

```
template<typename ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< ATraits, Metadata >::iterator
```

Child iterator.

Definition at line 320 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

4.663 `std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >` Struct Template Reference

## Public Types

- **typedef \_Distance** [difference\\_type](#)
- **typedef \_Category** [iterator\\_category](#)
- **typedef \_Pointer** [pointer](#)
- **typedef \_Reference** [reference](#)
- **typedef \_Tp** [value\\_type](#)

#### 4.663.1 Detailed Description

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference =
_Tp&>
struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
```

Common iterator class.

This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.

In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

Definition at line 127 of file `stl_iterator_base_types.h`.

#### 4.663.2 Member Typedef Documentation

##### 4.663.2.1 `difference_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵
_Tp*, typename _Reference = _Tp&>
typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_type
```

Distance between iterators is represented as this type.

Definition at line 134 of file `stl_iterator_base_types.h`.

##### 4.663.2.2 `iterator_category`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↵
_Tp*, typename _Reference = _Tp&>
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_category
```

One of the [tag types](#).

Definition at line 130 of file `stl_iterator_base_types.h`.

#### 4.663.2.3 `pointer`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
 typename _Reference = _Tp&>
typedef _Pointer std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file `stl_iterator_base_types.h`.

#### 4.663.2.4 `reference`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
 typename _Reference = _Tp&>
typedef _Reference std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value\_type.

Definition at line 138 of file `stl_iterator_base_types.h`.

#### 4.663.2.5 `value_type`

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*,
 typename _Reference = _Tp&>
typedef _Tp std::iterator<_Category, _Tp, _Distance, _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 4.664 `std::iterator_traits<_Iterator>` Struct Template Reference

Inherits `std::__iterator_traits<_Iterator, typename>`.

#### 4.664.1 Detailed Description

```
template<typename _Iterator>
struct std::iterator_traits< _Iterator >
```

Traits class for iterators.

This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the `Iterator` argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.

Definition at line 423 of file `cpp_type_traits.h`.

The documentation for this struct was generated from the following file:

- [cpp\\_type\\_traits.h](#)

#### 4.665 `std::iterator_traits< _Tp * >` Struct Template Reference

##### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef \_Tp \* **pointer**
- typedef \_Tp & **reference**
- typedef \_Tp **value\_type**

#### 4.665.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits< _Tp * >
```

Partial specialization for pointer types.

Definition at line 210 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

#### 4.666 `std::iterator_traits< const _Tp * >` Struct Template Reference

##### Public Types

- typedef ptrdiff\_t **difference\_type**
- typedef [random\\_access\\_iterator\\_tag](#) **iterator\_category**
- typedef const \_Tp \* **pointer**
- typedef const \_Tp & **reference**
- typedef \_Tp **value\_type**



## 4.666.1 Detailed Description

```
template<typename _Tp>
struct std::iterator_traits< const _Tp * >
```

Partial specialization for const pointer types.

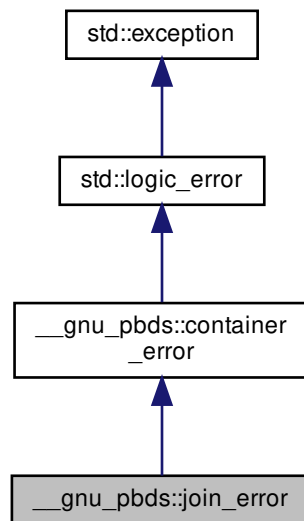
Definition at line 221 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.667 \_\_gnu\_pbds::join\_error Struct Reference

Inheritance diagram for \_\_gnu\_pbds::join\_error:



## Public Member Functions

- virtual const char \* [what](#) () const noexcept

## 4.667.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

Definition at line 70 of file exception.hpp.

#### 4.667.2 Member Function Documentation

##### 4.667.2.1 what()

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

#### 4.668 `__gnu_pbds::detail::left_child_next_sibling_heap`< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc > Class Template Reference

Inherits `Cmp_Fn`.

##### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `left_child_next_sibling_heap_const_iterator_< node, _Alloc >` **const\_iterator**
- typedef `__rebind_v::const_pointer` **const\_pointer**
- typedef `__rebind_v::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `const_iterator` **iterator**
- typedef `left_child_next_sibling_heap_node_< Value_Type, Node_Metadata, _Alloc >` **node**
- typedef `left_child_next_sibling_heap_node_point_const_iterator_< node, _Alloc >` **point\_const\_iterator**
- typedef `point_const_iterator` **point\_iterator**
- typedef `__rebind_v::pointer` **pointer**
- typedef `__rebind_v::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- `left_child_next_sibling_heap` (const Cmp\_Fn &)
- `left_child_next_sibling_heap` (const `left_child_next_sibling_heap` &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- Cmp\_Fn & `get_cmp_fn` ()
- const Cmp\_Fn & `get_cmp_fn` () const
- size\_type `max_size` () const
- size\_type `size` () const
- void `swap` (`left_child_next_sibling_heap`< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc > &)

### Protected Types

- typedef alloc\_traits::allocator\_type `node_allocator`
- typedef alloc\_traits::const\_pointer `node_const_pointer`
- typedef Node\_Metadata `node_metadata`
- typedef alloc\_traits::pointer `node_pointer`
- typedef `std::pair`< node\_pointer, node\_pointer > `node_pointer_pair`

### Protected Member Functions

- void `actual_erase_node` (node\_pointer)
- void `bubble_to_top` (node\_pointer)
- void `clear_imp` (node\_pointer)
- node\_pointer `get_new_node_for_insert` (const\_reference)
- template<typename Pred >  
node\_pointer `prune` (Pred)
- void `swap_with_parent` (node\_pointer, node\_pointer)
- void `to_linked_list` ()
- void `value_swap` (`left_child_next_sibling_heap` &)

### Static Protected Member Functions

- static void `make_child_of` (node\_pointer, node\_pointer)
- static node\_pointer `parent` (node\_pointer)

### Protected Attributes

- node\_pointer `m_p_root`
- size\_type `m_size`

## 4.668.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >
```

Base class for a basic heap.

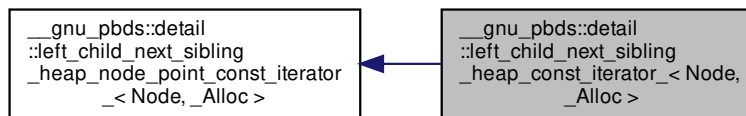
Definition at line 90 of file `left_child_next_sibling_heap_.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_.hpp](#)

4.669 `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >`:



## Public Types

- typedef `base_type::const_pointer` `const_pointer`
- typedef `base_type::const_reference` `const_reference`
- typedef `_Alloc::difference_type` `difference_type`
- typedef `std::forward_iterator_tag` `iterator_category`
- typedef `base_type::pointer` `pointer`
- typedef `base_type::reference` `reference`
- typedef `base_type::value_type` `value_type`

## Public Member Functions

- `left_child_next_sibling_heap_const_iterator_` (node\_pointer p\_nd)
- `left_child_next_sibling_heap_const_iterator_` ()
- `left_child_next_sibling_heap_const_iterator_` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other)
- bool `operator!=` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other) const
- bool `operator!=` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const
- `const_reference` `operator*` () const
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` & `operator++` ()
- `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` `operator++` (int)
- `const_pointer` `operator->` () const
- bool `operator==` (const `left_child_next_sibling_heap_const_iterator_< Node, _Alloc >` &other) const
- bool `operator==` (const `left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` &other) const

## Public Attributes

- node\_pointer `m_p_nd`

### 4.669.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

Definition at line 60 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

### 4.669.2 Member Typedef Documentation

#### 4.669.2.1 `const_pointer`

```
template<typename Node , typename _Alloc >
typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 81 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.2 `const_reference`

```
template<typename Node , typename _Alloc >
typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 87 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.3 `difference_type`

```
template<typename Node , typename _Alloc >
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
```

Difference type.

Definition at line 72 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.4 iterator\_category

```
template<typename Node , typename _Alloc >
typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

Definition at line 69 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.5 pointer

```
template<typename Node , typename _Alloc >
typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 78 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.6 reference

```
template<typename Node , typename _Alloc >
typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
```

Iterator's reference type.

Definition at line 84 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.2.7 value\_type

```
template<typename Node , typename _Alloc >
typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

Definition at line 75 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

### 4.669.3 Constructor & Destructor Documentation

## 4.669.3.1 left\_child\_next\_sibling\_heap\_const\_iterator\_() [1/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_co
() [inline]
```

Default constructor.

Definition at line 96 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

## 4.669.3.2 left\_child\_next\_sibling\_heap\_const\_iterator\_() [2/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_next_sibling_heap_co
(
 const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) [inline]
```

Copy constructor.

Definition at line 101 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

## 4.669.4 Member Function Documentation

## 4.669.4.1 operator!=(()) [1/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
 const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 111 of file left\_child\_next\_sibling\_heap\_/const\_iterator.hpp.

## 4.669.4.2 operator!=(()) [2/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
 const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

Definition at line 127 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 4.669.4.3 operator\*()

```
template<typename Node , typename _Alloc >
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline], [inherited]
```

Access.

Definition at line 114 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.669.4.4 operator->()

```
template<typename Node , typename _Alloc >
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline], [inherited]
```

Access.

Definition at line 106 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.669.4.5 operator==( ) [1/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
 const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content to a different iterator object.

Definition at line 106 of file `left_child_next_sibling_heap_/const_iterator.hpp`.

#### 4.669.4.6 operator==( ) [2/2]

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
 const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const [inline], [inherited]
```

Compares content to a different iterator object.

Definition at line 122 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/const\\_iterator.hpp](#)



## 4.670 `__gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >` Struct Template Reference

### Public Types

- typedef `_Metadata` **metadata\_type**
- typedef `rebind_traits<_Alloc, this_type>::pointer` **node\_pointer**
- typedef `_Alloc::size_type` **size\_type**
- typedef `_Value` **value\_type**

### Public Attributes

- `metadata_type` **m\_metadata**
- `node_pointer` **m\_p\_l\_child**
- `node_pointer` **m\_p\_next\_sibling**
- `node_pointer` **m\_p\_prev\_or\_parent**
- `value_type` **m\_value**

#### 4.670.1 Detailed Description

```
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_<_Value, _Metadata, _Alloc >
```

Node.

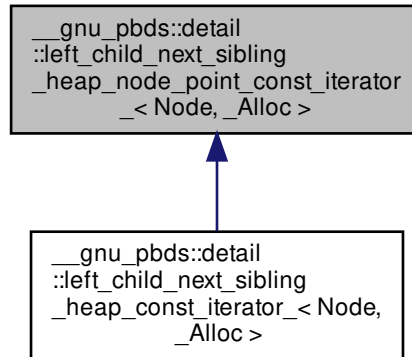
Definition at line 52 of file `left_child_next_sibling_heap_/node.hpp`.

The documentation for this struct was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/node.hpp](#)

#### 4.671 `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >`:



#### Public Types

- typedef `rebind_traits< _Alloc, value_type >::const_pointer` `const_pointer`
- typedef `rebind_traits< _Alloc, value_type >::const_reference` `const_reference`
- typedef `trivial_iterator_difference_type` `difference_type`
- typedef `trivial_iterator_tag` `iterator_category`
- typedef `rebind_traits< _Alloc, value_type >::pointer` `pointer`
- typedef `rebind_traits< _Alloc, value_type >::reference` `reference`
- typedef `Node::value_type` `value_type`

#### Public Member Functions

- `left_child_next_sibling_heap_node_point_const_iterator_` (`node_pointer p_nd`)
- `left_child_next_sibling_heap_node_point_const_iterator_` ()
- `left_child_next_sibling_heap_node_point_const_iterator_` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`)
- `bool operator!=` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`
- `const_reference operator*` () `const`
- `const_pointer operator->` () `const`
- `bool operator==` (`const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other`) `const`

#### Public Attributes

- `node_pointer` `m_p_nd`

## Protected Types

- typedef `rebind_traits<_Alloc, Node >::pointer` `node_pointer`

### 4.671.1 Detailed Description

```
template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >
```

Const point-type iterator.

Definition at line 61 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

### 4.671.2 Member Typedef Documentation

#### 4.671.2.1 `const_pointer`

```
template<typename Node , typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

Definition at line 81 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.2.2 `const_reference`

```
template<typename Node , typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::const_reference
```

Iterator's const reference type.

Definition at line 88 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.2.3 `difference_type`

```
template<typename Node , typename _Alloc >
typedef trivial_iterator_difference_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::difference_type
```

Difference type.

Definition at line 71 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.2.4 iterator\_category

```
template<typename Node , typename _Alloc >
typedef trivial_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

Definition at line 68 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 4.671.2.5 pointer

```
template<typename Node , typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_co
Node, _Alloc >::pointer
```

Iterator's pointer type.

Definition at line 77 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 4.671.2.6 reference

```
template<typename Node , typename _Alloc >
typedef rebind_traits<_Alloc, value_type>::reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_
Node, _Alloc >::reference
```

Iterator's reference type.

Definition at line 84 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

#### 4.671.2.7 value\_type

```
template<typename Node , typename _Alloc >
typedef Node::value_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

Definition at line 74 of file left\_child\_next\_sibling\_heap\_/point\_const\_iterator.hpp.

### 4.671.3 Constructor & Destructor Documentation

#### 4.671.3.1 `left_child_next_sibling_heap_node_point_const_iterator_()` [1/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ () [inline]
```

Default constructor.

Definition at line 96 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.3.2 `left_child_next_sibling_heap_node_point_const_iterator_()` [2/2]

```
template<typename Node , typename _Alloc >
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↔
::left_child_next_sibling_heap_node_point_const_iterator_ (
 const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
 other) [inline]
```

Copy constructor.

Definition at line 101 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

### 4.671.4 Member Function Documentation

#### 4.671.4.1 `operator!=(())`

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!=(
 const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
 other) const [inline]
```

Compares content (negatively) to a different iterator object.

Definition at line 127 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.4.2 `operator*()`

```
template<typename Node , typename _Alloc >
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const [inline]
```

Access.

Definition at line 114 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.4.3 operator->()

```
template<typename Node , typename _Alloc >
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const [inline]
```

Access.

Definition at line 106 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

#### 4.671.4.4 operator==()

```
template<typename Node , typename _Alloc >
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
 const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
 other) const [inline]
```

Compares content to a different iterator object.

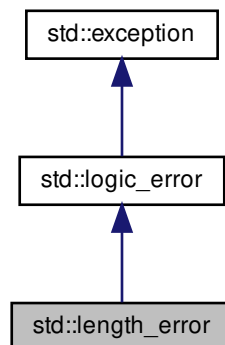
Definition at line 122 of file `left_child_next_sibling_heap_/point_const_iterator.hpp`.

The documentation for this class was generated from the following file:

- [left\\_child\\_next\\_sibling\\_heap\\_/point\\_const\\_iterator.hpp](#)

### 4.672 std::length\_error Class Reference

Inheritance diagram for `std::length_error`:



## Public Member Functions

- `length_error` (const [string](#) &\_\_arg) `_GLIBCXX_TXN_SAFE`
- `length_error` (const char \*) `_GLIBCXX_TXN_SAFE`
- `length_error` (const [length\\_error](#) &)=default
- `length_error` ([length\\_error](#) &&)=default
- `length_error` & `operator=` (const [length\\_error](#) &)=default
- `length_error` & `operator=` ([length\\_error](#) &&)=default
- virtual const char \* `what` () const noexcept

## 4.672.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a `basic_string` instance).

Definition at line 184 of file `stdexcept`.

## 4.672.2 Member Function Documentation

4.672.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

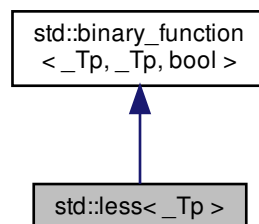
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.673 `std::less<_Tp>` Struct Template Reference

Inheritance diagram for `std::less<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 4.673.1 Detailed Description

```
template<typename _Tp>
struct std::less<_Tp >
```

One of the [comparison functors](#).

Definition at line 340 of file `stl_function.h`.

#### 4.673.2 Member Typedef Documentation

##### 4.673.2.1 first\_argument\_type

```
typedef _Tp std::binary_function<_Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.673.2.2 result\_type

```
typedef bool std::binary_function<_Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.



4.673.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.674 `std::less< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`__t`, `_Up` &&`__u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`__t`)<[std::forward](#)< `_Up` >(`__u`))) -> decltype([std::forward](#)< `_Tp` >(`__t`)< [std::forward](#)< `_Up` >(`__u`))
- template<typename `_Tp` , typename `_Up` >  
constexpr bool **operator()** (`_Tp` \*`__t`, `_Up` \*`__u`) const noexcept

## 4.674.1 Detailed Description

```
template<>
struct std::less< void >
```

One of the [comparison functors](#).

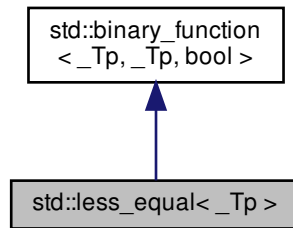
Definition at line 578 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.675 `std::less_equal<_Tp>` Struct Template Reference

Inheritance diagram for `std::less_equal<_Tp>`:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

##### 4.675.1 Detailed Description

```
template<typename _Tp>
struct std::less_equal<_Tp>
```

One of the [comparison functors](#).

Definition at line 346 of file `stl_function.h`.

##### 4.675.2 Member Typedef Documentation

4.675.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.675.2.2 `result_type`

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.675.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.676 `std::less_equal< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **`is_transparent`**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **`operator()`** (`_Tp` &&`__t`, `_Up` &&`__u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`__t`)<=[std::forward](#)< `_Up` >(`__u`))) -> decltype([std::forward](#)< `_Tp` >(`__t`)<=[std::forward](#)< `_Up` >(`__u`))
- template<typename `_Tp` , typename `_Up` >  
constexpr bool **`operator()`** (`_Tp` \*`__t`, `_Up` \*`__u`) const noexcept

#### 4.676.1 Detailed Description

```
template<>
struct std::less_equal< void >
```

One of the [comparison functors](#).

Definition at line 702 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.677 \_\_gnu\_cxx::limit\_condition::limit\_adjustor Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

##### Public Member Functions

- **limit\_adjustor** (const size\_t \_\_l)

#### 4.677.1 Detailed Description

Enter the nth condition.

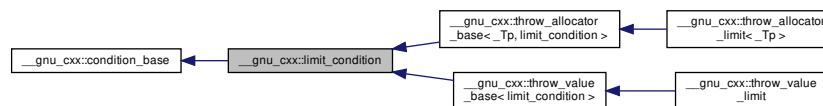
Definition at line 458 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.678 \_\_gnu\_cxx::limit\_condition Struct Reference

Inheritance diagram for `__gnu_cxx::limit_condition`:



## Classes

- struct [always\\_adjustor](#)
- struct [limit\\_adjustor](#)
- struct [never\\_adjustor](#)

## Static Public Member Functions

- static `size_t` & **count** ()
- static `size_t` & **limit** ()
- static void **set\_limit** (const `size_t` \_\_l)
- static void **throw\_conditionally** ()

## 4.678.1 Detailed Description

Base class for incremental control and throw.

Definition at line 428 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

4.679 `std::linear_congruential_engine<_UIntType, __a, __c, __m>` Class Template Reference

## Public Types

- typedef `_UIntType` [result\\_type](#)

## Public Member Functions

- [linear\\_congruential\\_engine](#) ()
- [linear\\_congruential\\_engine](#) ([result\\_type](#) \_\_s)
- template<typename `_Sseq`, typename `=_If_seed_seq<_Sseq>`>>  
[linear\\_congruential\\_engine](#) (`_Sseq` &\_\_q)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) [operator\(\)](#) ()
- template<typename `_Sseq`>  
auto [seed](#) (`_Sseq` &\_\_q) -> `_If_seed_seq<_Sseq>`
- void [seed](#) ([result\\_type](#) \_\_s=default\_seed)
- template<typename `_Sseq`>  
`_If_seed_seq<_Sseq>` [seed](#) (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr [result\\_type](#) [max](#) ()
- static constexpr [result\\_type](#) [min](#) ()

## Static Public Attributes

- static constexpr [result\\_type](#) **default\_seed**
- static constexpr [result\\_type](#) **increment**
- static constexpr [result\\_type](#) **modulus**
- static constexpr [result\\_type](#) **multiplier**

## Friends

- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)
- bool [operator==](#) (const [linear\\_congruential\\_engine](#) &\_\_lhs, const [linear\\_congruential\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, \_UIntType1 \_\_a1, \_UIntType1 \_\_c1, \_UIntType1 \_\_m1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::linear\\_congruential\\_engine](#)< \_UIntType1, \_\_a1, \_\_c1, \_\_m1 > &\_\_lcr)

### 4.679.1 Detailed Description

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >
```

A model of a linear congruential random number generator.

A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to `(__m-1)`. If the template parameter `__m` is 0, the modulus `__m` used is `std::numeric_limits<_UIntType>::max()` plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.

The size of the state is 1.

Definition at line 255 of file `random.h`.

### 4.679.2 Member Typedef Documentation

#### 4.679.2.1 `result_type`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

Definition at line 268 of file `random.h`.

## 4.679.3 Constructor &amp; Destructor Documentation

## 4.679.3.1 linear\_congruential\_engine() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine () [inline]
```

Constructs a linear\_congruential\_engine random number generator engine with seed 1.

Definition at line 282 of file random.h.

## 4.679.3.2 linear\_congruential\_engine() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
 result_type __s) [inline], [explicit]
```

Constructs a linear\_congruential\_engine random number generator engine with seed \_\_s. The default seed value is 1.

## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__s</code> | The initial seed value. |
|------------------|-------------------------|

Definition at line 293 of file random.h.

References std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::seed().

## 4.679.3.3 linear\_congruential\_engine() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
 _Sseq & __q) [inline], [explicit]
```

Constructs a linear\_congruential\_engine random number generator engine seeded from the seed sequence \_\_q.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Definition at line 304 of file random.h.

References `std::linear_congruential_engine<_UIntType, __a, __c, __m >::seed()`.

#### 4.679.4 Member Function Documentation

##### 4.679.4.1 `discard()`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine<_UIntType, __a, __c, __m >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 348 of file random.h.

##### 4.679.4.2 `max()`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m >::max ()
[inline], [static]
```

Gets the largest possible value in the output range.

Definition at line 341 of file random.h.

##### 4.679.4.3 `min()`

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine<_UIntType, __a, __c, __m >::min ()
[inline], [static]
```

Gets the smallest possible value in the output range.

The minimum depends on the `__c` parameter: if it is zero, the minimum generated must be  $> 0$ , otherwise 0 is allowed.

Definition at line 334 of file random.h.



## 4.679.4.4 operator()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() () [inline]
```

Gets the next random number in the sequence.

Definition at line 358 of file random.h.

## 4.679.4.5 seed() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq >
auto std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
 _Sseq & __q) -> _If_seed_seq<_Sseq>
```

Seeds the LCR engine with a value generated by \_\_q.

Definition at line 133 of file bits/random.tcc.


References std::\_\_lg().

## 4.679.4.6 seed() [2/3]

```
template<typename _UIntType , _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
 result_type __s = default_seed)
```

Reseeds the linear\_congruential\_engine random number generator engine sequence to the seed \_\_s.

## Parameters

|                                                                                         |               |
|-----------------------------------------------------------------------------------------|---------------|
|  __s | The new seed. |
|-----------------------------------------------------------------------------------------|---------------|

Seeds the LCR with integral value \_\_s, adjusted so that the ring identity is never a member of the convergence set.

Definition at line 117 of file bits/random.tcc.

Referenced by std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >::linear\_congruential\_engine().

**4.679.4.7 seed()** [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq >
_if_seed_seq<_Sseq> std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
 _Sseq & __q)
```

Reseeds the linear\_congruential\_engine random number generator engine sequence using values from the seed sequence \_\_q.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

**4.679.5 Friends And Related Function Documentation****4.679.5.1 operator<<**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
, typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)
[friend]
```

Writes the textual representation of the state x(i) of x to \_\_os.

**Parameters**

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__os</code>  | The output stream.                                      |
| <code>__lcr</code> | A % linear_congruential_engine random number generator. |

**Returns**

`__os`.

**4.679.5.2 operator==**

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
 const linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
 const linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs) [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

## Parameters

|                    |                                                             |
|--------------------|-------------------------------------------------------------|
| <code>__lhs</code> | A linear congruential random number generator object.       |
| <code>__rhs</code> | Another linear congruential random number generator object. |

## Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 376 of file random.h.

## 4.679.5.3 operator&gt;&gt;

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT
, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits > & __is,
 std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr) [friend]
```

Sets the state of the engine by reading its textual representation from `__is`.

The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments `_CharT` and `_Traits` were the same as those of `__is`.

## Parameters

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>__is</code>  | The input stream.                                       |
| <code>__lcr</code> | A % linear_congruential_engine random number generator. |

## Returns

`__is`.

## 4.679.6 Member Data Documentation

## 4.679.6.1 increment

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment [static]
```

An increment.

Definition at line 273 of file random.h.

#### 4.679.6.2 modulus

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus [static]
```

The modulus.

Definition at line 275 of file random.h.

#### 4.679.6.3 multiplier

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier [static]
```

The multiplier.

Definition at line 271 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.680 \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type > Class Template Reference

#### Public Types

- typedef Size\_Type **size\_type**

#### Public Member Functions

- void **swap** ([linear\\_probe\\_fn](#)< Size\_Type > &other)

#### Protected Member Functions

- size\_type [operator\(\)](#) (size\_type i) const

#### 4.680.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >
```

A probe sequence policy using fixed increments.

Definition at line 61 of file hash\_policy.hpp.

## 4.680.2 Member Function Documentation

## 4.680.2.1 operator()( )

```
template<typename Size_Type >
linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

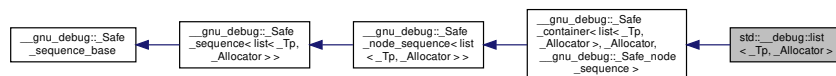
Definition at line 53 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 4.681 std::\_\_debug::list&lt; \_Tp, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::list< \_Tp, \_Allocator >:



## Public Types

- typedef \_Allocator **allocator\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_const\_iterator, list > **const\_iterator**
- typedef \_Base::const\_pointer **const\_pointer**
- typedef \_Base::const\_reference **const\_reference**
- typedef std::reverse\_iterator< const\_iterator > **const\_reverse\_iterator**
- typedef \_Base::difference\_type **difference\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator< \_Base\_iterator, list > **iterator**
- typedef \_Base::pointer **pointer**
- typedef \_Base::reference **reference**
- typedef std::reverse\_iterator< iterator > **reverse\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- **list** (const [list](#) &)=default
- **list** ([list](#) &&)=default
- **list** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- **list** (const [list](#) &\_\_x, const allocator\_type &\_\_a)
- **list** ([list](#) &&\_\_x, const allocator\_type &\_\_a)
- **list** (const \_Allocator &\_\_a) noexcept
- **list** (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- **list** (size\_type \_\_n, const \_Tp &\_\_value, const \_Allocator &\_\_a=\_Allocator())
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
**list** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Allocator &\_\_a=\_Allocator())
- **list** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- void **assign** ([initializer\\_list](#)< value\_type > \_\_l)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** (size\_type \_\_n, const \_Tp &\_\_t)
- reference **back** () noexcept
- const\_reference **back** () const noexcept
- [iterator](#) **begin** () noexcept
- [const\\_iterator](#) **begin** () const noexcept
- [const\\_iterator](#) **cbegin** () const noexcept
- [const\\_iterator](#) **cend** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- template<typename... \_Args>  
[iterator](#) **emplace** ([const\\_iterator](#) \_\_position, \_Args &&... \_\_args)
- [iterator](#) **end** () noexcept
- [const\\_iterator](#) **end** () const noexcept
- [iterator](#) **erase** ([const\\_iterator](#) \_\_position) noexcept
- [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last) noexcept
- reference **front** () noexcept
- const\_reference **front** () const noexcept
- [iterator](#) **insert** ([const\\_iterator](#) \_\_position, const \_Tp &\_\_x)
- [iterator](#) **insert** ([const\\_iterator](#) \_\_position, \_Tp &&\_\_x)
- [iterator](#) **insert** ([const\\_iterator](#) \_\_p, [initializer\\_list](#)< value\_type > \_\_l)
- [iterator](#) **insert** ([const\\_iterator](#) \_\_position, size\_type \_\_n, const \_Tp &\_\_x)
- template<class \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
[iterator](#) **insert** ([const\\_iterator](#) \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- void **merge** ([list](#) &&\_\_x)
- void **merge** ([list](#) &\_\_x)
- template<class \_Compare >  
void **merge** ([list](#) &&\_\_x, \_Compare \_\_comp)
- template<typename \_Compare >  
void **merge** ([list](#) &\_\_x, \_Compare \_\_comp)

- [list](#) & **operator=** (const [list](#) &)=default
- [list](#) & **operator=** ([list](#) &&)=default
- [list](#) & **operator=** ([initializer\\_list](#)< value\_type > \_\_l)
- void **pop\_back** () noexcept
- void **pop\_front** () noexcept
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- \_\_remove\_return\_type **remove** (const \_Tp &\_\_value)
- template<class \_Predicate >  
\_\_remove\_return\_type **remove\_if** (\_Predicate \_\_pred)
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **resize** (size\_type \_\_sz)
- void **resize** (size\_type \_\_sz, const \_Tp &\_\_c)
- void **sort** ()
- template<typename \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_pred)
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &&\_\_x) noexcept
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &\_\_x) noexcept
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &&\_\_x, [const\\_iterator](#) \_\_i) noexcept
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &\_\_x, [const\\_iterator](#) \_\_i) noexcept
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &&\_\_x, [const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last) noexcept
- void **splice** ([const\\_iterator](#) \_\_position, [list](#) &\_\_x, [const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last) noexcept
- void **swap** ([list](#) &\_\_x) noexcept(*/\*conditional \*/*)
- \_\_remove\_return\_type **unique** ()
- template<class \_BinaryPredicate >  
\_\_remove\_return\_type **unique** (\_BinaryPredicate \_\_binary\_pred)

#### Public Attributes

- \_Safe\_iterator\_base \* [\\_M\\_const\\_iterators](#)
- \_Safe\_iterator\_base \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

#### Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- \_\_gnu\_cxx::\_\_mutex & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- \_Safe\_container & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x) noexcept

#### Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT >  
class ::[\\_\\_gnu\\_debug::Safe\\_iterator](#)

#### 4.681.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::list<_Tp, _Allocator>
```

Class std::list with safety/checking/debug instrumentation.

Definition at line 36 of file debug/list.

#### 4.681.2 Member Function Documentation

##### 4.681.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

##### 4.681.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

##### 4.681.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().



#### 4.681.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.681.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< list< _Tp, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

#### 4.681.2.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.681.2.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.681.2.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< list< _Tp, _Allocator > >::_M_transfer_from_if (
 _Safe_sequence< list< _Tp, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file safe\_sequence.tcc.

### 4.681.3 Member Data Documentation

#### 4.681.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.681.3.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.681.3.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

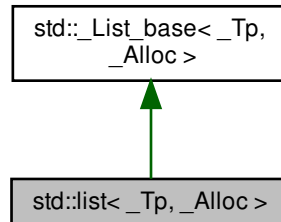
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/list](#)

## 4.682 std::list&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::list< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_List_const_iterator< _Tp >` **const\_iterator**
- typedef `_Tp_alloc_traits::const_pointer` **const\_pointer**
- typedef `_Tp_alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_List_iterator< _Tp >` **iterator**
- typedef `_Tp_alloc_traits::pointer` **pointer**
- typedef `_Tp_alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- `list()`=default
- `list(const allocator_type &__a)` noexcept
- `list(size_type __n, const allocator_type &__a=allocator_type())`
- `list(size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())`
- `list(const list &__x)`
- `list(list &&)=default`
- `list(initializer_list< value_type > __l, const allocator_type &__a=allocator_type())`
- `list(const list &__x, const allocator_type &__a)`
- `list(list &&__x, const allocator_type &__a)` noexcept(`_Node_alloc_traits::_S_always_equal()`)
- `template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>>`  
`list(_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `~list()`=default
- `void assign(size_type __n, const value_type &__val)`

- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (initializer_list< value_type > __l)`
- `reference back () noexcept`
- `const_reference back () const noexcept`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
`iterator emplace (const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_front (_Args &&... __args)`
- `bool empty () const noexcept`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `iterator erase (const_iterator __position) noexcept`
- `iterator erase (const_iterator __first, const_iterator __last) noexcept`
- `reference front () noexcept`
- `const_reference front () const noexcept`
- `allocator_type get_allocator () const noexcept`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `iterator insert (const_iterator __p, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const value_type &__x)`
- `template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `size_type max_size () const noexcept`
- `void merge (list &&__x)`
- `void merge (list &__x)`
- `template<typename _StrictWeakOrdering >`  
`void merge (list &&__x, _StrictWeakOrdering __comp)`
- `template<typename _StrictWeakOrdering >`  
`void merge (list &__x, _StrictWeakOrdering __comp)`
- `list & operator= (const list &__x)`
- `list & operator= (list &&__x) noexcept(_Node_alloc_traits::_S_nothrow_move())`
- `list & operator= (initializer_list< value_type > __l)`
- `void pop_back () noexcept`
- `void pop_front () noexcept`
- `void push_back (const value_type &__x)`
- `void push_back (value_type &&__x)`
- `void push_front (const value_type &__x)`
- `void push_front (value_type &&__x)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `__remove_return_type remove (const _Tp &__value)`

- template<typename \_Predicate >  
\_\_remove\_return\_type remove\_if (\_Predicate)
- reverse\_iterator rend () noexcept
- const\_reverse\_iterator rend () const noexcept
- void resize (size\_type \_\_new\_size)
- void resize (size\_type \_\_new\_size, const value\_type &\_\_x)
- void reverse () noexcept
- size\_type size () const noexcept
- void sort ()
- template<typename \_StrictWeakOrdering >  
void sort (\_StrictWeakOrdering)
- void splice (const\_iterator \_\_position, list &&\_\_x) noexcept
- void splice (const\_iterator \_\_position, list &\_\_x) noexcept
- void splice (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_i) noexcept
- void splice (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_i) noexcept
- void splice (const\_iterator \_\_position, list &&\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void splice (const\_iterator \_\_position, list &\_\_x, const\_iterator \_\_first, const\_iterator \_\_last) noexcept
- void swap (list &\_\_x) noexcept
- \_\_remove\_return\_type unique ()
- template<typename \_BinaryPredicate >  
\_\_remove\_return\_type unique (\_BinaryPredicate)

### Protected Types

- typedef \_List\_node< \_Tp > \_Node

### Protected Member Functions

- template<typename \_Integer >  
void \_M\_assign\_dispatch (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- template<typename \_InputIterator >  
void \_M\_assign\_dispatch (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- void \_M\_check\_equal\_allocators (list &\_\_x) noexcept
- void \_M\_clear () noexcept
- template<typename... \_Args>  
\_Node \* \_M\_create\_node (\_Args &&... \_\_args)
- void \_M\_dec\_size (size\_t)
- void \_M\_default\_append (size\_type \_\_n)
- void \_M\_default\_initialize (size\_type \_\_n)
- size\_t \_M\_distance (const void \*, const void \*) const
- void \_M\_erase (iterator \_\_position) noexcept
- void \_M\_fill\_assign (size\_type \_\_n, const value\_type &\_\_val)
- void \_M\_fill\_initialize (size\_type \_\_n, const value\_type &\_\_x)
- \_Node\_alloc\_traits::pointer \_M\_get\_node ()
- \_Node\_alloc\_traits::pointer \_M\_get\_node ()
- \_Node\_alloc\_type & \_M\_get\_Node\_allocator () noexcept
- const \_Node\_alloc\_type & \_M\_get\_Node\_allocator () const noexcept
- \_Node\_alloc\_type & \_M\_get\_Node\_allocator () noexcept
- const \_Node\_alloc\_type & \_M\_get\_Node\_allocator () const noexcept

- `size_t _M_get_size () const`
- `void _M_inc_size (size_t)`
- `void _M_init () noexcept`
- `template<typename _Integer >`  
`void _M_initialize_dispatch (_Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_initialize_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename... _Args>`  
`void _M_insert (iterator __position, _Args &&... __args)`
- `void _M_move_assign (list &&__x, true_type) noexcept`
- `void _M_move_assign (list &&__x, false_type)`
- `void _M_move_nodes (_List_base &&__x)`
- `size_t _M_node_count () const`
- `void _M_put_node (typename _Node_alloc_traits::pointer __p) noexcept`
- `void _M_put_node (typename _Node_alloc_traits::pointer __p) noexcept`
- `const_iterator _M_resize_pos (size_type &__new_size) const`
- `void _M_set_size (size_t)`
- `void _M_transfer (iterator __position, iterator __first, iterator __last)`

#### Static Protected Member Functions

- `static size_t _S_distance (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)`
- `static size_t _S_distance (const_iterator, const_iterator)`

#### Protected Attributes

- `_List_impl _M_impl`

#### 4.682.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list< _Tp, _Alloc >
```

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

#### Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `at` and `operator[]`.

This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike `std::vector` and `std::deque`,

random-access iterators are not provided, so subscripting ( `[]` ) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.

Also unlike the other standard containers, `std::list` provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

A couple points on memory allocation for `list<Tp>`:

First, we never actually allocate a `Tp`, we allocate `List_node<Tp>`'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from `list<X,Alloc1>` are spliced into `list<X,Alloc2>`, destroying the memory of the second list is a valid operation, i.e., `Alloc1` giveth and `Alloc2` taketh away.

Second, a list conceptually represented as

A <--> B <--> C <--> D

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private `list::iterator` pointing to *D*, not to *A*! To get to the head of the list, we start at the tail and move forward by one. When this member iterator's `next/previous` pointers refer to itself, the list is empty.

Definition at line 556 of file `stl_list.h`.

## 4.682.2 Constructor & Destructor Documentation

### 4.682.2.1 list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list () [default]
```

Creates a list with no elements.

### 4.682.2.2 list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 const allocator_type & __a) [inline], [explicit], [noexcept]
```

Creates a list with no elements.

#### Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|



Definition at line 683 of file `stl_list.h`.

**4.682.2.3 list()** [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a list with default constructed elements.

**Parameters**

|                                                                                              |                                             |
|----------------------------------------------------------------------------------------------|---------------------------------------------|
|  <b>__n</b> | The number of elements to initially create. |
|  <b>__a</b> | An allocator object.                        |

This constructor fills the list with **\_\_n** default constructed elements.

Definition at line 696 of file stl\_list.h.

**4.682.2.4 list()** [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline]
```

Creates a list with copies of an exemplar element.

**Parameters**

|                |                                             |
|----------------|---------------------------------------------|
| <b>__n</b>     | The number of elements to initially create. |
| <b>__value</b> | An element to copy.                         |
| <b>__a</b>     | An allocator object.                        |

This constructor fills the list with **\_\_n** copies of **\_\_value**.

Definition at line 708 of file stl\_list.h.

**4.682.2.5 list()** [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list<_Tp, _Alloc >::list (
 const list<_Tp, _Alloc > & __x) [inline]
```

List copy constructor.



## Parameters

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__x</code> | A list of identical element and allocator types. |
|------------------|--------------------------------------------------|

The newly-created list uses a copy of the allocation object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 735 of file `stl_list.h`.

## 4.682.2.6 list() [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 list< _Tp, _Alloc > &&) [default]
```

List move constructor.

The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

## 4.682.2.7 list() [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
 initializer_list< value_type > __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from an `initializer_list`.

## Parameters

|                  |                                                               |
|------------------|---------------------------------------------------------------|
| <code>__l</code> | An <code>initializer_list</code> of <code>value_type</code> . |
| <code>__a</code> | An allocator object.                                          |

Create a list consisting of copies of the elements in the `initializer_list __l`. This is linear in `__l.size()`.

Definition at line 758 of file `stl_list.h`.

## 4.682.2.8 list() [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
```

```
std::list< _Tp, _Alloc >::list (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a list from a range.

#### Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__first</code> | An input iterator.   |
| <code>__last</code>  | An input iterator.   |
| <code>__a</code>     | An allocator object. |

Create a list consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first, __last)`).

Definition at line 803 of file `stl_list.h`.

#### 4.682.2.9 ~list()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~~list () [default]
```

No explicit dtor needed as the `_Base` dtor takes care of things. The `_Base` dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

#### 4.682.3 Member Function Documentation

##### 4.682.3.1 \_M\_create\_node()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
_Node* std::list< _Tp, _Alloc >::_M_create_node (
 _Args &&... __args) [inline], [protected]
```

#### Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>__args</code> | An instance of user data. |
|---------------------|---------------------------|

Allocates space for a new node and constructs a copy of `__args` in it.

Definition at line 632 of file `stl_list.h`.

### 4.682.3.2 assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
 size_type __n,
 const value_type & __val) [inline]
```

Assigns a given value to a list.

#### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a list with `__n` copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

Definition at line 889 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::operator=()`.

### 4.682.3.3 assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>>
void std::list< _Tp, _Alloc >::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a list.

#### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a list with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

Definition at line 908 of file `stl_list.h`.

**4.682.3.4 assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
 initializer_list< value_type > __l) [inline]
```

Assigns an initializer\_list to a list.

**Parameters**

|   |                                    |
|---|------------------------------------|
| ↩ | An initializer_list of value_type. |
| ↩ |                                    |
| ↩ |                                    |
| ↩ |                                    |
| / |                                    |

Replace the contents of the list with copies of the elements in the initializer\_list \_\_l. This is linear in \_\_l.size().

Definition at line 930 of file stl\_list.h.

**4.682.3.5 back()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

Definition at line 1130 of file stl\_list.h.

**4.682.3.6 back()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::back () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

Definition at line 1142 of file stl\_list.h.

#### 4.682.3.7 begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 945 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::crend(), std::list< \_\_inp, \_\_rebind\_inp >::front(), std::list< \_\_inp, \_\_rebind\_inp >::insert(), std::list< \_\_inp, \_\_rebind\_inp >::list(), std::list< \_\_inp, \_\_rebind\_inp >::merge(), std::operator<(), std::list< \_\_inp, \_\_rebind\_inp >::operator=(), std::operator==( ), std::list< \_\_inp, \_\_rebind\_inp >::pop\_front(), std::list< \_\_inp, \_\_rebind\_inp >::push\_front(), std::list< \_\_inp, \_\_rebind\_inp >::rend(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

#### 4.682.3.8 begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 954 of file stl\_list.h.

#### 4.682.3.9 cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

Definition at line 1018 of file stl\_list.h.

#### 4.682.3.10 cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 1027 of file stl\_list.h.

#### 4.682.3.11 clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1498 of file `stl_list.h`.

#### 4.682.3.12 crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 1036 of file `stl_list.h`.

#### 4.682.3.13 crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 1045 of file `stl_list.h`.

#### 4.682.3.14 emplace()

```
template<typename _Tp , typename _Alloc >
template<typename... _Args>
list< _Tp, _Alloc >::iterator list::emplace (
 const_iterator __position,
 _Args &&... __args)
```

Constructs object in list before specified iterator.

##### Parameters

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the list. |
| <code>__args</code>     | Arguments.                                   |

**Returns**

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 90 of file list.tcc.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert().

**4.682.3.15 empty()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the list is empty. (Thus begin() would equal end().)

Definition at line 1055 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::insert(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

**4.682.3.16 end()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 963 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::back(), std::list< \_\_inp, \_\_rebind\_inp >::cbegin(), std::list< \_\_inp, \_\_rebind\_inp >::list(), std::list< \_\_inp, \_\_rebind\_inp >::merge(), std::operator<(), std::list< \_\_inp, \_\_rebind\_inp >::operator=(), std::operator==( ), std::list< \_\_inp, \_\_rebind\_inp >::push\_back(), std::list< \_\_inp, \_\_rebind\_inp >::rbegin(), and std::list< \_\_inp, \_\_rebind\_inp >::splice().

**4.682.3.17 end()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

Definition at line 972 of file stl\_list.h.

**4.682.3.18 erase()** [1/2]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::iterator list::erase (
 const_iterator __position) [noexcept]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the list by one.

Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 152 of file `list.tcc`.

Referenced by `std::list< __inp, __rebind_inp >::erase()`.

**4.682.3.19 erase()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline], [noexcept]
```

Remove a range of elements.

**Parameters**

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

**Returns**

An iterator pointing to the element pointed to by *last* prior to erasing (or `end()`).

This function will erase the elements in the range `[first,last)` and shorten the list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1456 of file `stl_list.h`.



**4.682.3.20 front()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

Definition at line 1114 of file stl\_list.h.

**4.682.3.21 front()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

Definition at line 1122 of file stl\_list.h.

**4.682.3.22 get\_allocator()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 936 of file stl\_list.h.

**4.682.3.23 insert()** [1/5]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 const value_type & __x)
```

Inserts given value into list before specified iterator.

**Parameters**

|                         |                                 |
|-------------------------|---------------------------------|
| <code>__position</code> | A const_iterator into the list. |
| <code>__x</code>        | Data to be inserted.            |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 103 of file list.tcc.

Referenced by `std::list< __inp, __rebind_inp >::insert()`.

**4.682.3.24 insert()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts given rvalue into list before specified iterator.

**Parameters**

|                         |                                 |
|-------------------------|---------------------------------|
| <code>__position</code> | A const_iterator into the list. |
| <code>__x</code>        | Data to be inserted.            |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1308 of file stl\_list.h.

**4.682.3.25 insert()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
 const_iterator __p,
 initializer_list< value_type > __l) [inline]
```

Inserts the contents of an initializer\_list into list before specified const\_iterator.

## Parameters

|                                 |                                    |
|---------------------------------|------------------------------------|
| $\leftarrow$<br><code>_p</code> | A const_iterator into the list.    |
| $\leftarrow$<br><code>_l</code> | An initializer_list of value_type. |

## Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the initializer\_list `l` into the list before the location specified by `p`.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 1327 of file `stl_list.h`.

## 4.682.3.26 insert() [4/5]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x)
```

Inserts a number of copies of given data into the list.

## Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the list.    |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

## Returns

An iterator pointing to the first element inserted (or `__position`).

This function will insert a specified number of copies of the given data before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 118 of file `list.tcc`.

**4.682.3.27 insert()** [5/5]

```
template<typename _Tp , typename _Alloc >
template<typename _InputIterator , typename >
list< _Tp, _Alloc >::iterator list::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last)
```

Inserts a range into the list.

**Parameters**

|                         |                                 |
|-------------------------|---------------------------------|
| <code>__position</code> | A const_iterator into the list. |
| <code>__first</code>    | An input iterator.              |
| <code>__last</code>     | An input iterator.              |

**Returns**

An iterator pointing to the first element inserted (or `__position`).

This function will insert copies of the data in the range *[first,last)* into the list before the location specified by *position*.

This operation is linear in the number of elements inserted and does not invalidate iterators and references.

Definition at line 134 of file list.tcc.

**4.682.3.28 max\_size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the size() of the largest possible list.

Definition at line 1065 of file stl\_list.h.

**4.682.3.29 merge()** [1/2]

```
template<typename _Tp , typename _Alloc >
void list::merge (
 list< _Tp, _Alloc > && __x)
```

Merge sorted lists.

## Parameters

|                  |                       |
|------------------|-----------------------|
| <code>_↵</code>  | Sorted list to merge. |
| <code>__x</code> |                       |

Assumes that both `__x` and this list are sorted according to `operator<()`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equal.

Definition at line 394 of file list.tcc.

## 4.682.3.30 merge() [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _StrictWeakOrdering >
void list::merge (
 list< _Tp, _Alloc > && __x,
 _StrictWeakOrdering __comp)
```

Merge sorted lists according to comparison function.

## Template Parameters

|                                   |                                          |
|-----------------------------------|------------------------------------------|
| <code>__StrictWeakOrdering</code> | Comparison function defining sort order. |
|-----------------------------------|------------------------------------------|

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__x</code>    | Sorted list to merge. |
| <code>__comp</code> | Comparison functor.   |

Assumes that both `__x` and this list are sorted according to `StrictWeakOrdering`. Merges elements of `__x` into this list in sorted order, leaving `__x` empty when complete. Elements in this list precede elements in `__x` that are equivalent according to `StrictWeakOrdering()`.

Definition at line 441 of file list.tcc.

## 4.682.3.31 operator=() [1/3]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc > & list::operator= (
 const list< _Tp, _Alloc > & __x)
```

List assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | A list of identical element and allocator types. |
| <code>__x</code> |                                                  |

All the elements of `__x` are copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 268 of file `list.tcc`.

**4.682.3.32 operator=()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (
 list< _Tp, _Alloc > && __x) [inline], [noexcept]
```

List move assignment operator.

**Parameters**

|                  |                                                  |
|------------------|--------------------------------------------------|
| <code>__l</code> | A list of identical element and allocator types. |
| <code>__x</code> |                                                  |

The contents of `__x` are moved into this list (without copying).

Afterwards `__x` is a valid, but unspecified list

Whether the allocator is moved depends on the allocator traits.

Definition at line 853 of file `stl_list.h`.

**4.682.3.33 operator=()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list& std::list< _Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

List initializer list assignment operator.

## Parameters

|   |                                    |
|---|------------------------------------|
| ↵ | An initializer_list of value_type. |
| ↵ |                                    |
| ↵ |                                    |
| ↵ |                                    |
| / |                                    |

Replace the contents of the list with copies of the elements in the initializer\_list \_\_l. This is linear in l.size().

Definition at line 871 of file stl\_list.h.

## 4.682.3.34 pop\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop\_back() is called.

Definition at line 1246 of file stl\_list.h.

## 4.682.3.35 pop\_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front () [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop\_front() is called.

Definition at line 1197 of file stl\_list.h.

## 4.682.3.36 push\_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_back (
 const value_type & __x) [inline]
```

Add data to the end of the list.

**Parameters**

|                         |                   |
|-------------------------|-------------------|
| $\leftrightarrow$<br>_X | Data to be added. |
|-------------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1211 of file `stl_list.h`.

**4.682.3.37 push\_front()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
 const value_type & __x) [inline]
```

Add data to the front of the list.

**Parameters**

|                         |                   |
|-------------------------|-------------------|
| $\leftrightarrow$<br>_X | Data to be added. |
|-------------------------|-------------------|

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

Definition at line 1161 of file `stl_list.h`.

**4.682.3.38 rbegin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 981 of file `stl_list.h`.

**4.682.3.39 rbegin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

Definition at line 990 of file `stl_list.h`.



## 4.682.3.40 remove()

```
template<typename _Tp, typename _Alloc >
list< _Tp, _Alloc >::__remove_return_type list::remove (
 const _Tp & __value)
```

Remove all elements equal to *value*.

## Parameters

|                      |                      |
|----------------------|----------------------|
| <code>__value</code> | The value to remove. |
|----------------------|----------------------|

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 332 of file list.tcc.

## 4.682.3.41 remove\_if()

```
template<typename _Tp , typename _Alloc >
template<typename _Predicate >
list< _Tp, _Alloc >::__remove_return_type list::remove_if (
 _Predicate __pred)
```

Remove all elements satisfying a predicate.

## Template Parameters

|                         |                                     |
|-------------------------|-------------------------------------|
| <code>_Predicate</code> | Unary predicate function or object. |
|-------------------------|-------------------------------------|

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 534 of file list.tcc.

## 4.682.3.42 rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 999 of file stl\_list.h.

**4.682.3.43** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

Definition at line 1008 of file `stl_list.h`.

**4.682.3.44** `resize()` [1/2]

```
template<typename _Tp , typename _Alloc >
void list::resize (
 size_type __new_size)
```

Resizes the list to the specified number of elements.

**Parameters**

|                         |                                             |
|-------------------------|---------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain. |
|-------------------------|---------------------------------------------|

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.

Definition at line 231 of file `list.tcc`.

**4.682.3.45** `resize()` [2/2]

```
template<typename _Tp , typename _Alloc >
void list::resize (
 size_type __new_size,
 const value_type & __x)
```

Resizes the list to the specified number of elements.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the list should contain.       |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.

Definition at line 243 of file `list.tcc`.

**4.682.3.46 reverse()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse () [inline], [noexcept]
```

Reverse the elements in list.

Reverse the order of elements in the list in linear time.

Definition at line 1798 of file stl\_list.h.

**4.682.3.47 size()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size () const [inline], [noexcept]
```

Returns the number of elements in the list.

Definition at line 1060 of file stl\_list.h.

Referenced by std::list< \_\_inp, \_\_rebind\_inp >::merge(), and std::operator==().

**4.682.3.48 sort()** [1/2]

```
template<typename _Tp , typename _Alloc >
void list::sort ()
```

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 487 of file list.tcc.

**4.682.3.49 sort()** [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _StrictWeakOrdering >
void list::sort (
 _StrictWeakOrdering __comp)
```

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

Definition at line 585 of file list.tcc.

**4.682.3.50 splice()** [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > && __x) [inline], [noexcept]
```

Insert contents of another list.

**Parameters**

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>__position</code> | Iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                       |

The elements of `__x` are inserted in constant time in front of the element referenced by `__position`. `__x` becomes an empty list.

Requires this `!= __x`.

Definition at line 1518 of file `stl_list.h`.

Referenced by `std::list< __inp, __rebind_inp >::splice()`.

**4.682.3.51 splice()** [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > && __x,
 const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

**Parameters**

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__i</code>        | Const_iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1553 of file `stl_list.h`.

**4.682.3.52 splice()** [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > & __x,
 const_iterator __i) [inline], [noexcept]
```

Insert element from another list.

## Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__i</code>        | Const_iterator referencing the element to move.          |

Removes the element in list `__x` referenced by `__i` and inserts it into the current list before `__position`.

Definition at line 1595 of file `stl_list.h`.

## 4.682.3.53 splice() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > && __x,
 const_iterator __first,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

## Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__first</code>    | Const_iterator referencing the start of range in x.      |
| <code>__last</code>     | Const_iterator referencing the end of range in x.        |

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first,__last)`.

Definition at line 1614 of file `stl_list.h`.

## 4.682.3.54 splice() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
 const_iterator __position,
 list< _Tp, _Alloc > & __x,
 const_iterator __first,
 const_iterator __last) [inline], [noexcept]
```

Insert range from another list.

## Parameters

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>__position</code> | Const_iterator referencing the element to insert before. |
| <code>__x</code>        | Source list.                                             |
| <code>__first</code>    | Const_iterator referencing the start of range in x.      |
| <code>__last</code>     | Const_iterator referencing the end of range in x.        |

Removes elements in the range `[__first,__last)` and inserts them before `__position` in constant time.

Undefined if `__position` is in `[__first,__last)`.

Definition at line 1664 of file `stl_list.h`.

4.682.3.55 `swap()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
 list< _Tp, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another list.

## Parameters

|                  |                                                 |
|------------------|-------------------------------------------------|
| <code>__x</code> | A list of the same element and allocator types. |
|------------------|-------------------------------------------------|

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(l1,l2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1478 of file `stl_list.h`.

4.682.3.56 `unique()` [1/2]

```
template<typename _Tp , typename _Alloc >
list< _Tp, _Alloc >::__remove_return_type list::unique ()
```

Remove consecutive duplicate elements.

For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 368 of file `list.tcc`.

4.683.3.57 `unique()` [2/2]

```
template<typename _Tp , typename _Alloc >
template<typename _BinaryPredicate >
list< _Tp, _Alloc >::__remove_return_type list::unique (
 _BinaryPredicate __binary_pred)
```

Remove consecutive elements satisfying a predicate.

## Template Parameters

|                               |                                      |
|-------------------------------|--------------------------------------|
| <code>_BinaryPredicate</code> | Binary predicate function or object. |
|-------------------------------|--------------------------------------|

For each consecutive set of elements  $[first, last)$  that satisfy  $predicate(first, i)$  where  $i$  is an iterator in  $[first, last)$ , remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 557 of file `list.tcc`.

The documentation for this class was generated from the following files:

- [stl\\_list.h](#)
- [list.tcc](#)

4.683 `__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >` Class Template Reference

Inherits type< Key, Mapped, \_Alloc, list\_update\_tag, \_\_gnu\_cxx::typelist::create2< Eq\_Fn, Update\_Policy >::type >.

## Public Types

- typedef [list\\_update\\_tag](#) **container\_category**
- typedef Eq\_Fn **eq\_fn**
- typedef Update\_Policy **update\_policy**

## Public Member Functions

- template<typename It >  
[list\\_update](#) (It first, It last)
- **list\_update** (const [list\\_update](#) &other)
- [list\\_update](#) & **operator=** (const [list\\_update](#) &other)
- void **swap** ([list\\_update](#) &other)

## 4.683.1 Detailed Description

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy =
detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
```

A list-update based associative container.

**Template Parameters**

|                      |                                                                                   |
|----------------------|-----------------------------------------------------------------------------------|
| <i>Key</i>           | Key type.                                                                         |
| <i>Mapped</i>        | Map type.                                                                         |
| <i>Eq_Fn</i>         | Equal functor.                                                                    |
| <i>Update_Policy</i> | Update policy, determines when an element will be moved to the front of the list. |
| <i>_Alloc</i>        | Allocator type.                                                                   |

Base is detail::lu\_map.

Definition at line 815 of file assoc\_container.hpp.

**4.683.2 Constructor & Destructor Documentation****4.683.2.1 list\_update()**

```
template<typename Key , typename Mapped , class Eq_Fn = typename detail::default_eq_fn<Key>↵
::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
 It first,
 It last) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

Definition at line 831 of file assoc\_container.hpp.

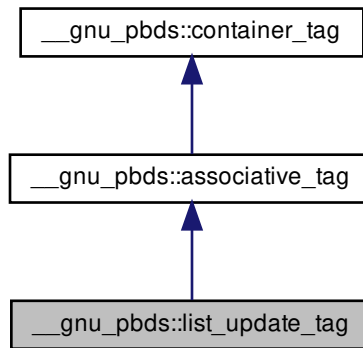
The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)



4.684 `__gnu_pbds::list_update_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::list_update_tag`:



## 4.684.1 Detailed Description

List-update.

Definition at line 168 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.685 `std::locale` Class Reference

## Classes

- class [facet](#)
- class [id](#)

## Public Types

- typedef int [category](#)

### Public Member Functions

- [locale](#) () throw ()
- [locale](#) (const [locale](#) &\_\_other) throw ()
- [locale](#) (const char \*\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const char \*\_\_s, [category](#) \_\_cat)
- [locale](#) (const [std::string](#) &\_\_s)
- [locale](#) (const [locale](#) &\_\_base, const [std::string](#) &\_\_s, [category](#) \_\_cat)
- [locale](#) (const [locale](#) &\_\_base, const [locale](#) &\_\_add, [category](#) \_\_cat)
- template<typename \_Facet >  
[locale](#) (const [locale](#) &\_\_other, \_Facet \*\_\_f)
- [~locale](#) () throw ()
- template<typename \_Facet >  
[locale combine](#) (const [locale](#) &\_\_other) const
- [\\_GLIBCXX\\_DEFAULT\\_ABI\\_TAG string name](#) () const
- bool [operator!=](#) (const [locale](#) &\_\_other) const throw ()
- template<typename \_CharT, typename \_Traits, typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_CharT, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_CharT, \_Traits, \_↵  
\_Alloc > &\_\_s2) const
- template<typename \_Char, typename \_Traits, typename \_Alloc >  
bool [operator\(\)](#) (const [basic\\_string](#)< \_Char, \_Traits, \_Alloc > &\_\_s1, const [basic\\_string](#)< \_Char, \_Traits, \_Alloc  
> &\_\_s2) const
- const [locale](#) & [operator=](#) (const [locale](#) &\_\_other) throw ()
- bool [operator==](#) (const [locale](#) &\_\_other) const throw ()

### Static Public Member Functions

- static const [locale](#) & [classic](#) ()
- static [locale global](#) (const [locale](#) &\_\_loc)

### Static Public Attributes

- static const [category none](#)
- static const [category ctype](#)
- static const [category numeric](#)
- static const [category collate](#)
- static const [category time](#)
- static const [category monetary](#)
- static const [category messages](#)
- static const [category all](#)

## Friends

- template<typename \_Cache >  
struct **\_\_use\_cache**
- class **\_Impl**
- class **facet**
- template<typename \_Facet >  
bool **has\_facet** (const **locale** &) throw ()
- template<typename \_Facet >  
const \_Facet & **use\_facet** (const **locale** &)

## 4.685.1 Detailed Description

Container class for localization functionality.

The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing.

Constructing C++ locales does not change the C library locale.

This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

Definition at line 62 of file locale\_classes.h.

## 4.685.2 Member Typedef Documentation

## 4.685.2.1 category

```
typedef int std::locale::category
```

Definition of locale::category.

Definition at line 67 of file locale\_classes.h.

## 4.685.3 Constructor &amp; Destructor Documentation

## 4.685.3.1 locale() [1/8]

```
std::locale::locale () throw ()
```

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

## 4.685.3.2 locale() [2/8]

```
std::locale::locale (
 const locale & __other) throw ()
```

Copy constructor.

Constructs a copy of *other*.

**Parameters**

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

**4.685.3.3 locale()** [3/8]

```
std::locale::locale (
 const char * __s) [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

**Parameters**

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

**Exceptions**

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

**4.685.3.4 locale()** [4/8]

```
std::locale::locale (
 const locale & __base,
 const char * __s,
 category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

**Parameters**

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

**Exceptions**

|                                 |                                                     |
|---------------------------------|-----------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is null or an undefined locale. |
|---------------------------------|-----------------------------------------------------|

## 4.685.3.5 locale() [5/8]

```
std::locale::locale (
 const std::string & __s) [inline], [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

## Parameters

|                  |                                  |
|------------------|----------------------------------|
| <code>__s</code> | Name of the locale to construct. |
|------------------|----------------------------------|

## Exceptions

|                                 |                                             |
|---------------------------------|---------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is an undefined locale. |
|---------------------------------|---------------------------------------------|

Definition at line 163 of file locale\_classes.h.

## 4.685.3.6 locale() [6/8]

```
std::locale::locale (
 const locale & __base,
 const std::string & __s,
 category __cat) [inline]
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If *base* is named, this locale instance will also be named.

## Parameters

|                     |                                                                      |
|---------------------|----------------------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                                  |
| <code>__s</code>    | Name of the locale to use facets from.                               |
| <code>__cat</code>  | Set of categories defining the facets to use from <code>__s</code> . |

## Exceptions

|                                 |                                             |
|---------------------------------|---------------------------------------------|
| <code>std::runtime_error</code> | if <code>__s</code> is an undefined locale. |
|---------------------------------|---------------------------------------------|

Definition at line 177 of file locale\_classes.h.

**4.685.3.7 locale()** [7/8]

```
std::locale::locale (
 const locale & __base,
 const locale & __add,
 category __cat)
```

Construct locale with facets from another locale.

Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

**Parameters**

|                     |                                                        |
|---------------------|--------------------------------------------------------|
| <code>__base</code> | The locale to copy.                                    |
| <code>__add</code>  | The locale to use facets from.                         |
| <code>__cat</code>  | Set of categories defining the facets to use from add. |

**4.685.3.8 locale()** [8/8]

```
template<typename _Facet >
std::locale::locale (
 const locale & __other,
 _Facet * __f)
```

Construct locale with another facet.

Constructs a copy of the locale *\_\_other*. The facet *\_\_f* is added to *\_\_other*, replacing an existing facet of type *Facet* if there is one. If *\_\_f* is null, this locale is a copy of *\_\_other*.

**Parameters**

|                      |                      |
|----------------------|----------------------|
| <code>__other</code> | The locale to copy.  |
| <code>__f</code>     | The facet to add in. |

Definition at line 45 of file locale\_classes.tcc.

**4.685.3.9 ~locale()**

```
std::locale::~~locale () throw ()
```

Locale destructor.

## 4.685.4 Member Function Documentation

## 4.685.4.1 classic()

```
static const locale& std::locale::classic () [static]
```

Return reference to the C locale.

## 4.685.4.2 combine()

```
template<typename _Facet >
locale std::locale::combine (
 const locale & __other) const
```

Construct locale with another facet.

Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type `Facet` from the locale *other* into the new locale.

## Template Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <code>_Facet</code> | The facet type to copy from other |
|---------------------|-----------------------------------|

## Parameters

|                      |                          |
|----------------------|--------------------------|
| <code>__other</code> | The locale to copy from. |
|----------------------|--------------------------|

## Returns

Newly constructed locale.

## Exceptions

|                                 |                                                                    |
|---------------------------------|--------------------------------------------------------------------|
| <code>std::runtime_error</code> | if <code>__other</code> has no facet of type <code>_Facet</code> . |
|---------------------------------|--------------------------------------------------------------------|

Definition at line 63 of file `locale_classes.tcc`.

## 4.685.4.3 global()

```
static locale std::locale::global (
 const locale & __loc) [static]
```

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call `std::setlocale(LC_ALL, loc.name())`.

#### Parameters

|                    |                                |
|--------------------|--------------------------------|
| <code>__loc</code> | The new locale to make global. |
|--------------------|--------------------------------|

#### Returns

Copy of the old global locale.

#### 4.685.4.4 `name()`

```
_GLIBCXX_DEFAULT_ABI_TAG string std::locale::name () const
```

Return locale name.

#### Returns

Locale name or "\*" if unnamed.

#### 4.685.4.5 `operator!=( )`

```
bool std::locale::operator!= (
 const locale & __other) const throw () [inline]
```

Locale inequality.

#### Parameters

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

#### Returns

`! (*this == __other)`

Definition at line 265 of file `locale_classes.h`.

References `operator==( )`.



## 4.685.4.6 operator()

```
template<typename _Char , typename _Traits , typename _Alloc >
bool std::locale::operator() (
 const basic_string< _Char, _Traits, _Alloc > & __s1,
 const basic_string< _Char, _Traits, _Alloc > & __s2) const
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector *v* of strings could be sorted according to locale *loc* by doing:

```
std::sort(v.begin(), v.end(), loc);
```

## Parameters

|                   |                           |
|-------------------|---------------------------|
| <code>__s1</code> | First string to compare.  |
| <code>__s2</code> | Second string to compare. |

## Returns

True if `collate<_Char>` facet compares `__s1 < __s2`, else false.

## 4.685.4.7 operator=()

```
const locale& std::locale::operator= (
 const locale & __other) throw ()
```

Assignment operator.

Set this locale to be a copy of *other*.

## Parameters

|                      |                     |
|----------------------|---------------------|
| <code>__other</code> | The locale to copy. |
|----------------------|---------------------|

## Returns

A reference to this locale.

## 4.685.4.8 operator==()

```
bool std::locale::operator== (
 const locale & __other) const throw ()
```

Locale equality.

**Parameters**

|                      |                                |
|----------------------|--------------------------------|
| <code>__other</code> | The locale to compare against. |
|----------------------|--------------------------------|

**Returns**

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

Referenced by operator!=().

**4.685.5 Friends And Related Function Documentation****4.685.5.1 has\_facet**

```
template<typename _Facet >
bool has_facet (
 const locale &) throw () [friend]
```

Test for the presence of a facet.

has\_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

**Template Parameters**

|                     |                                         |
|---------------------|-----------------------------------------|
| <code>_Facet</code> | The facet type to test the presence of. |
|---------------------|-----------------------------------------|

**Parameters**

|                    |                     |
|--------------------|---------------------|
| <code>__loc</code> | The locale to test. |
|--------------------|---------------------|

**Returns**

true if `__loc` contains a facet of type `_Facet`, else false.

Definition at line 104 of file locale\_classes.tcc.

**4.685.5.2 use\_facet**

```
template<typename _Facet >
const _Facet& use_facet (
 const locale &) [friend]
```

Return a facet.

`use_facet` looks for and returns a reference to a facet of type `Facet` where `Facet` is the template parameter. If `has_facet(locale)` is true, there is a suitable facet to return. It throws `std::bad_cast` if the locale doesn't contain a facet of type `Facet`.

#### Template Parameters

|                     |                           |
|---------------------|---------------------------|
| <code>_Facet</code> | The facet type to access. |
|---------------------|---------------------------|

#### Parameters

|                    |                    |
|--------------------|--------------------|
| <code>__loc</code> | The locale to use. |
|--------------------|--------------------|

#### Returns

Reference to facet of type `Facet`.

#### Exceptions

|                            |                                                                             |
|----------------------------|-----------------------------------------------------------------------------|
| <code>std::bad_cast</code> | if <code>__loc</code> doesn't contain a facet of type <code>_Facet</code> . |
|----------------------------|-----------------------------------------------------------------------------|

Definition at line 132 of file `locale_classes.tcc`.

### 4.685.6 Member Data Documentation

#### 4.685.6.1 all

```
const category std::locale::all [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 105 of file `locale_classes.h`.

#### 4.685.6.2 collate

```
const category std::locale::collate [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 101 of file `locale_classes.h`.

#### 4.685.6.3 ctype

```
const category std::locale::ctype [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 99 of file `locale_classes.h`.

#### 4.685.6.4 messages

```
const category std::locale::messages [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 104 of file `locale_classes.h`.

#### 4.685.6.5 monetary

```
const category std::locale::monetary [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 103 of file `locale_classes.h`.

#### 4.685.6.6 none

```
const category std::locale::none [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 98 of file `locale_classes.h`.

#### 4.685.6.7 numeric

```
const category std::locale::numeric [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 100 of file `locale_classes.h`.

#### 4.685.6.8 time

```
const category std::locale::time [static]
```

Category values.

The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match `_S_facet_categories` definition in `locale.cc`

Definition at line 102 of file `locale_classes.h`.

The documentation for this class was generated from the following files:

- [locale\\_classes.h](#)
- [locale\\_classes.tcc](#)

## 4.686 std::lock\_guard< \_Mutex > Class Template Reference

### Public Types

- typedef \_Mutex **mutex\_type**

### Public Member Functions

- **lock\_guard** (mutex\_type &\_\_m)
- **lock\_guard** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- **lock\_guard** (const [lock\\_guard](#) &)=delete
- [lock\\_guard](#) & **operator=** (const [lock\\_guard](#) &)=delete

### 4.686.1 Detailed Description

```
template<typename _Mutex>
class std::lock_guard< _Mutex >
```

A simple scoped lock type.

A lock\_guard controls mutex ownership within a scope, releasing ownership in the destructor.

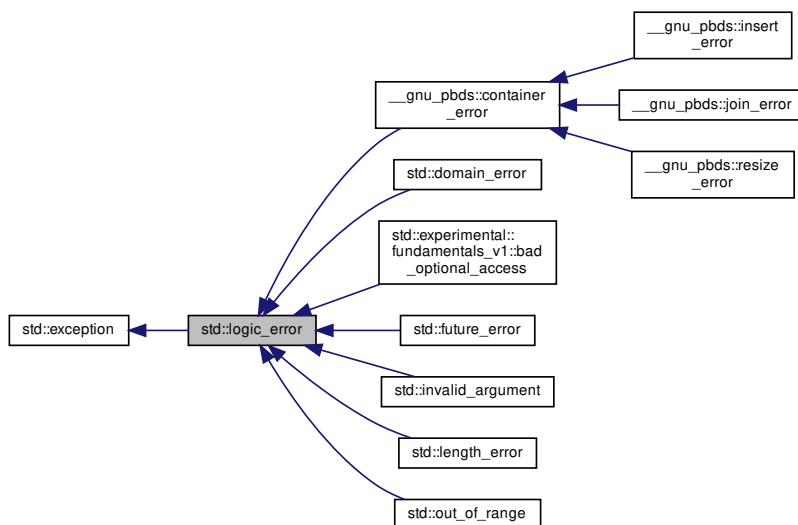
Definition at line 153 of file std\_mutex.h.

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 4.687 std::logic\_error Class Reference

Inheritance diagram for std::logic\_error:



## Public Member Functions

- [logic\\_error](#) (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- [logic\\_error](#) (const char \*) \_GLIBCXX\_TXN\_SAFE
- [logic\\_error](#) ([logic\\_error](#) &&) noexcept
- [logic\\_error](#) (const [logic\\_error](#) &)=default
- [logic\\_error](#) & [operator=](#) ([logic\\_error](#) &&) noexcept
- [logic\\_error](#) & [operator=](#) (const [logic\\_error](#) &)=default
- virtual const char \* [what](#) () const noexcept

### 4.687.1 Detailed Description

One of two subclasses of exception.

Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

Definition at line 113 of file stdexcept.

### 4.687.2 Constructor & Destructor Documentation

#### 4.687.2.1 [logic\\_error](#)()

```
std::logic_error::logic_error (
 const string & __arg) [explicit]
```

Takes a character string describing the error.

### 4.687.3 Member Function Documentation

#### 4.687.3.1 [what](#)()

```
virtual const char* std::logic_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

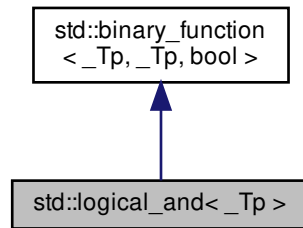
Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

#### 4.688 `std::logical_and< _Tp >` Struct Template Reference

Inheritance diagram for `std::logical_and< _Tp >`:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

##### 4.688.1 Detailed Description

```
template<typename _Tp>
struct std::logical_and< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 775 of file `stl_function.h`.

##### 4.688.2 Member Typedef Documentation



## 4.688.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 4.688.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 4.688.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.689 std::logical\_and&lt; void &gt; Struct Template Reference

## Public Types

- typedef \_\_is\_transparent **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) &&std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) &&std::forward< \_Up >(\_\_u))

#### 4.689.1 Detailed Description

```
template<>
struct std::logical_and< void >
```

One of the [Boolean operations functors](#).

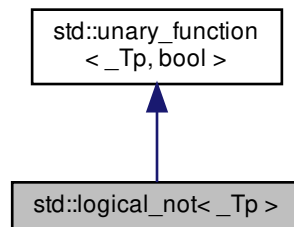
Definition at line 817 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.690 std::logical\_not< \_Tp > Struct Template Reference

Inheritance diagram for `std::logical_not< _Tp >`:



##### Public Types

- typedef `_Tp` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

##### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`) `const`

#### 4.690.1 Detailed Description

```
template<typename _Tp>
struct std::logical_not< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 781 of file `stl_function.h`.

## 4.690.2 Member Typedef Documentation

4.690.2.1 `argument_type`

```
typedef _Tp std::unary_function< _Tp , bool >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

4.690.2.2 `result_type`

```
typedef bool std::unary_function< _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.691 `std::logical_not< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` >  
constexpr auto **operator()** (`_Tp` &&`__t`) const noexcept(noexcept(![std::forward](#)< `_Tp` >(`__t`))) -> decltype(![std::forward](#)< `_Tp` >(`__t`))

## 4.691.1 Detailed Description

```
template<>
struct std::logical_not< void >
```

One of the [Boolean operations functors](#).

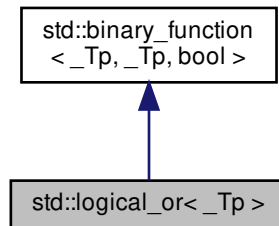
Definition at line 847 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.692 std::logical\_or< \_Tp > Struct Template Reference

Inheritance diagram for std::logical\_or< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 4.692.1 Detailed Description

```
template<typename _Tp>
struct std::logical_or< _Tp >
```

One of the [Boolean operations functors](#).

Definition at line 778 of file `stl_function.h`.

#### 4.692.2 Member Typedef Documentation

4.692.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.692.2.2 `result_type`

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.692.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.693 `std::logical_or< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t`←  
t)||`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`)||`std::forward`< `_Up` >(`_u`))

#### 4.693.1 Detailed Description

```
template<>
struct std::logical_or< void >
```

One of the [Boolean operations functors](#).

Definition at line 832 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.694 `std::lognormal_distribution<_RealType>` Class Template Reference

##### Classes

- struct [param\\_type](#)

##### Public Types

- typedef `_RealType` [result\\_type](#)

##### Public Member Functions

- **lognormal\_distribution** (`_RealType __m, _RealType __s=_RealType(1)`)
- **lognormal\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng`)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **generate** (`_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- `_RealType m` () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) **operator()** (`_UniformRandomNumberGenerator &__urng, const param\_type &__p`)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()
- `_RealType s` () const

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_ostream<\_CharT, \_Traits> & operator<< (std::basic\_ostream<\_CharT, \_Traits> &\_\_os, const std::lognormal\_distribution<\_RealType1> &\_\_x)
- bool operator== (const lognormal\_distribution &\_\_d1, const lognormal\_distribution &\_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits>  
std::basic\_istream<\_CharT, \_Traits> & operator>> (std::basic\_istream<\_CharT, \_Traits> &\_\_is, std::lognormal\_distribution<\_RealType1> &\_\_x)

## 4.694.1 Detailed Description

```
template<typename _RealType = double>
class std::lognormal_distribution<_RealType>
```

A lognormal\_distribution random number distribution.

The formula for the normal probability mass function is

$$p(x|m, s) = \frac{1}{sx\sqrt{2\pi}} \exp - \frac{(\ln x - m)^2}{2s^2}$$

Definition at line 2191 of file random.h.

## 4.694.2 Member Typedef Documentation

## 4.694.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::lognormal_distribution<_RealType>::result_type
```

The type of the range of the distribution.

Definition at line 2194 of file random.h.

## 4.694.3 Member Function Documentation

## 4.694.3.1 m()

```
template<typename _RealType = double>
_RealType std::lognormal_distribution<_RealType>::m () const [inline]
```

Definition at line 2256 of file random.h.

#### 4.694.3.2 max()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2289 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

#### 4.694.3.3 min()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2282 of file random.h.

#### 4.694.3.4 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::lognormal_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 2297 of file random.h.

#### 4.694.3.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::lognormal_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2267 of file random.h.

#### 4.694.3.6 param() [2/2]

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.



## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2275 of file `random.h`.

4.694.3.7 `reset()`

```
template<typename _RealType = double>
void std::lognormal_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

Definition at line 2249 of file `random.h`.

References `std::normal_distribution<_RealType>::reset()`.

## 4.694.4 Friends And Related Function Documentation

4.694.4.1 `operator<<`

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::lognormal_distribution<_RealType1> & __x) [friend]
```

Inserts a `lognormal_distribution` random number distribution `__x` into the output stream `__os`.

## Parameters

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                 |
| <code>__x</code>  | A <code>lognormal_distribution</code> random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

#### 4.694.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
 const lognormal_distribution< _RealType > & __d1,
 const lognormal_distribution< _RealType > & __d2) [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 2334 of file random.h.

#### 4.694.4.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::lognormal_distribution< _RealType1 > & __x) [friend]
```

Extracts a `lognormal_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                      |
| <code>__x</code>  | A <code>lognormal_distribution</code> random number generator engine. |

##### Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.695 `__gnu_pbds::detail::lu_counter_metadata< Size_Type >` Class Template Reference

##### Public Types

- typedef `Size_Type` **size\_type**

## Friends

- class **lu\_counter\_policy\_base**< **size\_type** >

## 4.695.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
```

A list-update metadata type that moves elements to the front of the list based on the counter algorithm.

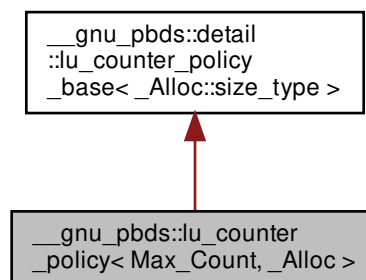
Definition at line 51 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 4.696 \_\_gnu\_pbds::lu\_counter\_policy&lt; Max\_Count, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >:



## Public Types

- enum { **max\_count** }
- typedef `_Alloc` **allocator\_type**
- typedef `detail::rebind_traits< _Alloc, metadata_type >::reference` **metadata\_reference**
- typedef `detail::lu_counter_metadata< size_type >` **metadata\_type**
- typedef `allocator_type::size_type` **size\_type**

### Public Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) ([metadata\\_reference](#) r\_data) const

### Private Member Functions

- [lu\\_counter\\_metadata](#)< size\_type > **operator()** (size\_type max\_size) const
- bool **operator()** ([Metadata\\_Reference](#) r\_data, size\_type m\_max\_count) const

## 4.696.1 Detailed Description

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
```

A list-update policy that moves elements to the front of the list based on the counter algorithm.

Definition at line 91 of file `list_update_policy.hpp`.

## 4.696.2 Member Typedef Documentation

### 4.696.2.1 metadata\_reference

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc, metadata_type>::reference __gnu_pbds::lu_counter_policy<
Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

Definition at line 114 of file `list_update_policy.hpp`.

### 4.696.2.2 metadata\_type

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

Definition at line 106 of file `list_update_policy.hpp`.

## 4.696.3 Member Enumeration Documentation

### 4.696.3.1 anonymous enum

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

## Enumerator

|                        |                                                                                                |
|------------------------|------------------------------------------------------------------------------------------------|
| <code>max_count</code> | When some element is accessed this number of times, it will be moved to the front of the list. |
|------------------------|------------------------------------------------------------------------------------------------|

Definition at line 98 of file `list_update_policy.hpp`.

## 4.696.4 Member Function Documentation

4.696.4.1 `operator()()` [1/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const [inline]
```

Creates a metadata object.

Definition at line 118 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

4.696.4.2 `operator()()` [2/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (
 metadata_reference r_data) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

Definition at line 124 of file `list_update_policy.hpp`.

References `__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::max_count`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

4.697 `__gnu_pbds::detail::lu_counter_policy_base< Size_Type >` Class Template Reference

## Protected Types

- typedef `Size_Type` **size\_type**

## Protected Member Functions

- [lu\\_counter\\_metadata](#)< size\_type > **operator()** (size\_type max\_size) const
- template<typename Metadata\_Reference >  
bool **operator()** (Metadata\_Reference r\_data, size\_type m\_max\_count) const

## 4.697.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >
```

Base class for list-update counter policy.

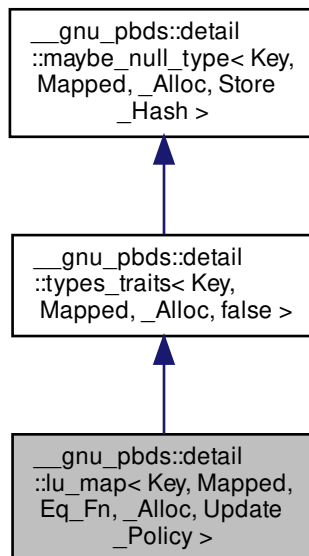
Definition at line 46 of file `lu_counter_metadata.hpp`.

The documentation for this class was generated from the following file:

- [lu\\_counter\\_metadata.hpp](#)

## 4.698 \_\_gnu\_pbds::detail::lu\_map&lt; Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy &gt; Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `Eq_Fn` **eq\_fn**
- typedef `iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `point_const_iterator` **point\_const\_iterator**
- typedef `point_iterator` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `Update_Policy::metadata_type` **update\_metadata**
- typedef `Update_Policy` **update\_policy**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **lu\_map** (const `lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)
- `template<typename It >`  
**lu\_map** (It, It)
- `iterator` **begin** ()
- `const_iterator` **begin** () const
- `void` **clear** ()
- `bool` **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- `bool` **erase** (key\_const\_reference)
- `template<typename Pred >`  
`size_type` **erase\_if** (Pred)
- `point_iterator` **find** (key\_const\_reference r\_key)
- `point_const_iterator` **find** (key\_const\_reference r\_key) const
- `std::pair< point_iterator, bool >` **insert** (const\_reference)
- `size_type` **max\_size** () const
- `mapped_reference` **operator[]** (key\_const\_reference r\_key)
- `size_type` **size** () const
- `void` **swap** (`lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >` &)

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Member Functions

- template<typename It >  
void **copy\_from\_range** (It, It)

### Friends

- class **const\_iterator\_**
- class **iterator\_**

### 4.698.1 Detailed Description

```
template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >
```

list-based (with updates) associative container. Skip to the lu, my darling.

Definition at line 91 of file lu\_map\_.hpp.

The documentation for this class was generated from the following file:

- [lu\\_map\\_.hpp](#)

### 4.699 \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc > Class Template Reference

#### Public Types

- typedef \_Alloc **allocator\_type**
- typedef [detail::rebind\\_traits](#)< \_Alloc, [metadata\\_type](#) >::reference [metadata\\_reference](#)
- typedef [null\\_type](#) [metadata\\_type](#)

#### Public Member Functions

- [metadata\\_type](#) [operator\(\)](#) () const
- bool [operator\(\)](#) ([metadata\\_reference](#) r\_metadata) const



#### 4.699.1 Detailed Description

```
template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy<_Alloc>
```

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

Definition at line 58 of file list\_update\_policy.hpp.

#### 4.699.2 Member Typedef Documentation

##### 4.699.2.1 metadata\_reference

```
template<typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc, metadata_type>::reference __gnu_pbds::lu_move_to_front_policy<
_Alloc>::metadata_reference
```

Reference to metadata on which this functor operates.

Definition at line 69 of file list\_update\_policy.hpp.

##### 4.699.2.2 metadata\_type

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::metadata_type
```

Metadata on which this functor operates.

Definition at line 64 of file list\_update\_policy.hpp.

#### 4.699.3 Member Function Documentation

##### 4.699.3.1 operator>() [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy<_Alloc>::operator() () const [inline]
```

Creates a metadata object.

Definition at line 73 of file list\_update\_policy.hpp.

#### 4.699.3.2 `operator()` [2/2]

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() (
 metadata_reference r_metadata) const [inline]
```

Decides whether a metadata object should be moved to the front of the list.

Definition at line 79 of file `list_update_policy.hpp`.

The documentation for this class was generated from the following file:

- [list\\_update\\_policy.hpp](#)

### 4.700 `std::make_signed< _Tp >` Struct Template Reference

#### Public Types

- typedef `__make_signed_selector< _Tp >::__type` **type**

#### 4.700.1 Detailed Description

```
template<typename _Tp>
struct std::make_signed< _Tp >
```

`make_signed`

Definition at line 1951 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

### 4.701 `std::make_unsigned< _Tp >` Struct Template Reference

#### Public Types

- typedef `__make_unsigned_selector< _Tp >::__type` **type**

## 4.701.1 Detailed Description

```
template<typename _Tp>
struct std::make_unsigned< _Tp >
```

make\_unsigned

Definition at line 1825 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.702 `__gnu_cxx::malloc_allocator<_Tp>` Class Template Reference

## Public Types

- typedef const \_Tp \* **const\_pointer**
- typedef const \_Tp & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef \_Tp \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef \_Tp & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- constexpr **malloc\_allocator** (const [malloc\\_allocator](#) &) noexcept
- template<typename \_Tp1 >  
constexpr **malloc\_allocator** (const [malloc\\_allocator](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- \_Tp \* **allocate** (size\_type \_\_n, const void \*==0)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args) noexcept([std::is\\_nothrow\\_constructible](#)<\_Up, \_Args... >::value)
- void **deallocate** (\_Tp \*\_\_p, size\_type)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p) noexcept([std::is\\_nothrow\\_destructible](#)<\_Up >::value)
- size\_type **max\_size** () const noexcept

## Friends

- template<typename \_Up >  
constexpr bool **operator!=** (const [malloc\\_allocator](#) &, const [malloc\\_allocator](#)<\_Up > &) noexcept
- template<typename \_Up >  
constexpr bool **operator==** (const [malloc\\_allocator](#) &, const [malloc\\_allocator](#)<\_Up > &) noexcept

#### 4.702.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::malloc_allocator< _Tp >
```

An allocator that uses malloc.

This is precisely the allocator defined in the C++ Standard.

- all allocation calls malloc
- all deallocation calls free

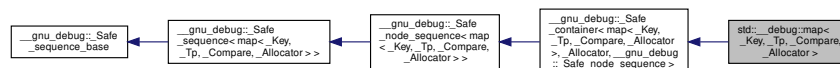
Definition at line 54 of file malloc\_allocator.h.

The documentation for this class was generated from the following file:

- [malloc\\_allocator.h](#)

#### 4.703 std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator > Class Template Reference

Inheritance diagram for std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >:



#### Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, map >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, map >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **map** (const [map](#) &)=default
- **map** ([map](#) &&)=default
- **map** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_c=[\\_Compare](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- **map** (const [allocator\\_type](#) &\_\_a)
- **map** (const [map](#) &\_\_m, const [allocator\\_type](#) &\_\_a)
- **map** ([map](#) &&\_\_m, const [allocator\\_type](#) &\_\_a) noexcept(noexcept([\\_Base](#)([std::move](#)(\_\_m.\_M\_base()), \_\_a)))
- **map** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- [template](#)<typename [\\_InputIterator](#) >  
**map** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [allocator\\_type](#) &\_\_a)
- **map** (const [\\_Base](#) &\_\_x)
- **map** (const [\\_Compare](#) &\_\_comp, const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- [template](#)<typename [\\_InputIterator](#) >  
**map** ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [\\_Allocator](#) &\_\_a=[\\_Allocator](#)())
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_invalidate\\_if](#) ([\\_Predicate](#) \_\_pred)
- void [\\_M\\_swap](#) ([\\_Safe\\_container](#) &\_\_x) noexcept
- void [\\_M\\_transfer\\_from\\_if](#) ([\\_Safe\\_sequence](#) &\_\_from, [\\_Predicate](#) \_\_pred)
- [iterator](#) [begin](#) () noexcept
- [const\\_iterator](#) [begin](#) () const noexcept
- [const\\_iterator](#) [cbegin](#) () const noexcept
- [const\\_iterator](#) [cend](#) () const noexcept
- void [clear](#) () noexcept
- [const\\_reverse\\_iterator](#) [crbegin](#) () const noexcept
- [const\\_reverse\\_iterator](#) [crend](#) () const noexcept
- [template](#)<typename... [\\_Args](#)>  
[std::pair](#)< [iterator](#), bool > [emplace](#) ([\\_Args](#) &&... \_\_args)
- [template](#)<typename... [\\_Args](#)>  
[iterator](#) [emplace\\_hint](#) (const [iterator](#) \_\_pos, [\\_Args](#) &&... \_\_args)
- [iterator](#) [end](#) () noexcept
- const [iterator](#) [end](#) () const noexcept
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const [key\\_type](#) &\_\_x)
- [template](#)<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const [\\_Kt](#) &\_\_x)
- [std::pair](#)< const [iterator](#), const [iterator](#) > [equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- [template](#)<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[std::pair](#)< const [iterator](#), const [iterator](#) > [equal\\_range](#) (const [\\_Kt](#) &\_\_x) const
- [iterator](#) [erase](#) (const [iterator](#) \_\_position)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11](#) [iterator](#) [erase](#) ([iterator](#) \_\_position)
- [size\\_type](#) [erase](#) (const [key\\_type](#) &\_\_x)
- [iterator](#) [erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [iterator](#) [find](#) (const [key\\_type](#) &\_\_x)
- [template](#)<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
[iterator](#) [find](#) (const [\\_Kt](#) &\_\_x)
- const [iterator](#) [find](#) (const [key\\_type](#) &\_\_x) const
- [template](#)<typename [\\_Kt](#), typename [\\_Req](#) = typename [\\_\\_has\\_is\\_transparent](#)<[\\_Compare](#), [\\_Kt](#)>::type>  
const [iterator](#) [find](#) (const [\\_Kt](#) &\_\_x) const

- `std::pair< iterator, bool > insert (const value_type &__x)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> std::pair< iterator, bool > insert (_Pair &&__x)`
- `void insert (std::initializer_list< value_type > __list)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator insert (const_iterator __position, _Pair &&__x)`
- `template<typename _InputIterator > void insert (_InputIterator __first, _InputIterator __last)`
- `iterator lower_bound (const key_type &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> iterator lower_bound (const _Kt &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> const_iterator lower_bound (const _Kt &__x) const`
- `map & operator= (const map &)=default`
- `map & operator= (map &&)=default`
- `map & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void swap (map &__x) noexcept(/*conditional */)`
- `iterator upper_bound (const key_type &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> iterator upper_bound (const _Kt &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type> const_iterator upper_bound (const _Kt &__x) const`

## Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

## Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular ()`
- `_Safe_container & _M_safe () noexcept`
- `void _M_swap (_Safe_sequence_base &__x) noexcept`

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
`class ::__gnu_debug::_Safe_iterator`

## 4.703.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >
```

Class std::map with safety/checking/debug instrumentation.

Definition at line 44 of file map.h.

## 4.703.2 Member Function Documentation

## 4.703.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

## 4.703.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

## 4.703.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.703.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.703.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 4.703.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.703.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.703.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if
(
 _Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.



### 4.703.3 Member Data Documentation

#### 4.703.3.1 \_M\_const\_iterators

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.703.3.2 \_M\_iterators

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.703.3.3 \_M\_version

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [map.h](#)

## 4.704 `std::map< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

### Public Member Functions

- `map()`=default
- `map(const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `map(const map &)=default`
- `map(map &&)=default`
- `map(initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `map(const allocator_type &__a)`
- `map(const map &__m, const allocator_type &__a)`
- `map(map && __m, const allocator_type &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && _Alloc_traits::_S_always_equal())`
- `map(initializer_list< value_type > __l, const allocator_type &__a)`
- template<typename `_InputIterator` >  
`map(_InputIterator __first, _InputIterator __last, const allocator_type &__a)`
- template<typename `_InputIterator` >  
`map(_InputIterator __first, _InputIterator __last)`
- template<typename `_InputIterator` >  
`map(_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())`
- `~map()`=default
- `mapped_type &at(const key_type &__k)`
- `const mapped_type &at(const key_type &__k) const`
- `iterator begin()` noexcept
- `const_iterator begin()` const noexcept
- `const_iterator cbegin()` const noexcept
- `const_iterator cend()` const noexcept
- `void clear()` noexcept
- `const_reverse_iterator crbegin()` const noexcept
- `const_reverse_iterator crend()` const noexcept

- template<typename... \_Args>  
std::pair< iterator, bool > **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - iterator **end** () noexcept
  - const\_iterator **end** () const noexcept
  - size\_type **erase** (const key\_type &\_\_x)
  - iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type **get\_allocator** () const noexcept
  - void **insert** (std::initializer\_list< value\_type > \_\_list)
  - template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - key\_compare **key\_comp** () const
  - size\_type **max\_size** () const noexcept
  - map & **operator=** (const map &)=default
  - map & **operator=** (map &&)=default
  - map & **operator=** (initializer\_list< value\_type > \_\_l)
  - mapped\_type & **operator[]** (const key\_type &\_\_k)
  - mapped\_type & **operator[]** (key\_type &&\_\_k)
  - reverse\_iterator **rbegin** () noexcept
  - const\_reverse\_iterator **rbegin** () const noexcept
  - reverse\_iterator **rend** () noexcept
  - const\_reverse\_iterator **rend** () const noexcept
  - size\_type **size** () const noexcept
  - void **swap** (map &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare **value\_comp** () const
- 
- std::pair< iterator, bool > **insert** (const value\_type &\_\_x)
  - std::pair< iterator, bool > **insert** (value\_type &&\_\_x)
  - template<typename \_Pair >  
\_\_enable\_if\_t< is\_constructible< value\_type, \_Pair >::value, pair< iterator, bool > > **insert** (\_Pair &&\_\_x)
- 
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_Pair >  
\_\_enable\_if\_t< is\_constructible< value\_type, \_Pair >::value, iterator > **insert** (const\_iterator \_\_position, \_Pair &&\_\_x)
- 
- iterator **erase** (const\_iterator \_\_position)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator **erase** (iterator \_\_position)

- iterator `find` (const key\_type &\_\_x)  
• template<typename \_Kt >  
  auto `find` (const \_Kt &\_\_x) -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- const\_iterator `find` (const key\_type &\_\_x) const  
• template<typename \_Kt >  
  auto `find` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- size\_type `count` (const key\_type &\_\_x) const  
• template<typename \_Kt >  
  auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- iterator `lower_bound` (const key\_type &\_\_x)  
• template<typename \_Kt >  
  auto `lower_bound` (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- const\_iterator `lower_bound` (const key\_type &\_\_x) const  
• template<typename \_Kt >  
  auto `lower_bound` (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- iterator `upper_bound` (const key\_type &\_\_x)  
• template<typename \_Kt >  
  auto `upper_bound` (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- const\_iterator `upper_bound` (const key\_type &\_\_x) const  
• template<typename \_Kt >  
  auto `upper_bound` (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- `std::pair`< iterator, iterator > `equal_range` (const key\_type &\_\_x)  
• template<typename \_Kt >  
  auto `equal_range` (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
- `std::pair`< const\_iterator, const\_iterator > `equal_range` (const key\_type &\_\_x) const  
• template<typename \_Kt >  
  auto `equal_range` (const \_Kt &\_\_x) const -> decltype(pair< const\_iterator, const\_iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))

## Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator< (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator== (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)`

## 4.704.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp> >>>
class std::map< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Template Parameters

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                |
| <code>_Tp</code>      | Type of mapped objects.                                                             |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .        |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys). For a `map<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Maps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 100 of file `stl_map.h`.

## 4.704.2 Constructor &amp; Destructor Documentation

4.704.2.1 `map()` [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>>
std::map< _Key, _Tp, _Compare, _Alloc >::map () [default]
```

Default constructor creates no elements.

#### 4.704.2.2 map() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a map with no elements.

##### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Definition at line 194 of file `stl_map.h`.

#### 4.704.2.3 map() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const map< _Key, _Tp, _Compare, _Alloc > &) [default]
```

Map copy constructor.

Whether the allocator is copied depends on the allocator traits.

#### 4.704.2.4 map() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 map< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Map move constructor.

The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

#### 4.704.2.5 map() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a map from an `initializer_list`.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__l</code>    | An initializer_list. |
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Create a map consisting of copies of the elements in the initializer\_list `__l`. This is linear in N if the range is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 228 of file `stl_map.h`.

## 4.704.2.6 map() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 236 of file `stl_map.h`.

## 4.704.2.7 map() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 const map< _Key, _Tp, _Compare, _Alloc > & __m,
 const allocator_type & __a) [inline]
```

Allocator-extended copy constructor.

Definition at line 240 of file `stl_map.h`.

## 4.704.2.8 map() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 map< _Key, _Tp, _Compare, _Alloc > && __m,
 const allocator_type & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 244 of file `stl_map.h`.

**4.704.2.9 map()** [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initialier-list constructor.

Definition at line 250 of file stl\_map.h.

**4.704.2.10 map()** [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

Definition at line 256 of file stl\_map.h.

**4.704.2.11 map()** [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a map from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 273 of file stl\_map.h.



## 4.704.2.12 map() [12/12]

```

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::map< _Key, _Tp, _Compare, _Alloc >::map (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]

```

Builds a map from a range.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a map consisting of copies of the elements from [`__first`,`__last`). This is linear in *N* if the range is already sorted, and *N*log*N* otherwise (where *N* is distance(`__first`,`__last`)).

Definition at line 290 of file `stl_map.h`.

## 4.704.2.13 ~map()

```

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::~~map () [default]

```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 4.704.3 Member Function Documentation

## 4.704.3.1 at()

```

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type& std::map< _Key, _Tp, _Compare, _Alloc >::at (
 const key_type & __k) [inline]

```

Access to map data.

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

**Returns**

A reference to the data whose key is equivalent to `__k`, if such a data is present in the map.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 537 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc>::end()`, `std::map<_Key, _Tp, _Compare, _Alloc>::key_comp()`, and `std::map<_Key, _Tp, _Compare, _Alloc>::lower_bound()`.

**4.704.3.2 `begin()`** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 356 of file `stl_map.h`.

**4.704.3.3 `begin()`** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
const_iterator std::map<_Key, _Tp, _Compare, _Alloc>::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 365 of file `stl_map.h`.

#### 4.704.3.4 cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 429 of file stl\_map.h.

#### 4.704.3.5 cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 438 of file stl\_map.h.

#### 4.704.3.6 clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.



Definition at line 1133 of file stl\_map.h.

#### 4.704.3.7 count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

**Parameters**

|                                                                                   |                                          |
|-----------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
|  |                                          |

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).



Definition at line 1215 of file stl\_map.h.

**4.704.3.8 count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::count (
 const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements with given key.

**Parameters**

|                                                                                     |                                          |
|-------------------------------------------------------------------------------------|------------------------------------------|
|  | Key of (key, value) pairs to be located. |
|  |                                          |

**Returns**

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

Definition at line 1221 of file stl\_map.h.

**4.704.3.9 crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 447 of file stl\_map.h.

## 4.704.3.10 crend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 456 of file stl\_map.h.

## 4.704.3.11 emplace()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
std::pair<iterator, bool> std::map< _Key, _Tp, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

## Parameters

|                     |                                                                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see std::piecewise_construct for passing arguments to each part of the pair constructor). |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 576 of file stl\_map.h.

## 4.704.3.12 emplace\_hint()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

## Parameters

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 606 of file `stl_map.h`.

4.704.3.13 `empty()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map<_Key, _Tp, _Compare, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the map is empty. (Thus `begin()` would equal `end()`.)

Definition at line 465 of file `stl_map.h`.

4.704.3.14 `end()` [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 374 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::at()`.

## 4.704.3.15 end() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Definition at line 383 of file stl\_map.h.

## 4.704.3.16 equal\_range() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, iterator> std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1352 of file stl\_map.h.

## 4.704.3.17 equal\_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↵</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1358 of file `stl_map.h`.

**4.704.3.18 equal\_range()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<const_iterator, const_iterator> std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↵</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

**Returns**

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1381 of file `stl_map.h`.



## 4.704.3.19 equal\_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator>(_M_t._M_↵
equal_range_tr(__x))) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                 |                                          |
|-----------------|------------------------------------------|
| <code>↵</code>  | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

Definition at line 1387 of file stl\_map.h.

## 4.704.3.20 erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a map.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1031 of file `stl_map.h`.

#### 4.704.3.21 `erase()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from a map.

##### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

##### Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1037 of file `stl_map.h`.

#### 4.704.3.22 `erase()` [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1068 of file stl\_map.h.

**4.704.3.23 erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a map.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1088 of file stl\_map.h.

**4.704.3.24 find()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a map.

**Parameters**

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1169 of file `stl_map.h`.

**4.704.3.25 find()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
 const _Kt & __x) -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

**Parameters**

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1175 of file `stl_map.h`.

**4.704.3.26 find()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a map.

## Parameters

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

## Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1194 of file `stl_map.h`.

## 4.704.3.27 find() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a map.

## Parameters

|                 |                                         |
|-----------------|-----------------------------------------|
| <code>_↵</code> | Key of (key, value) pair to be located. |
| <code>_X</code> |                                         |

## Returns

Read-only (constant) iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1200 of file `stl_map.h`.

## 4.704.3.28 get\_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 346 of file `stl_map.h`.

**4.704.3.29** `insert()` [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 803 of file `stl_map.h`.

Referenced by `std::map<_Key, _Tp, _Compare, _Alloc >::insert()`.

**4.704.3.30** `insert()` [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, bool> std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

**Parameters**

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 810 of file stl\_map.h.

References `std::move()`.

#### 4.704.3.31 insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, pair<iterator, bool> > std::map< _↵
Key, _Tp, _Compare, _Alloc >::insert (
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

##### Parameters

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

##### Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.

Insertion requires logarithmic time.

Definition at line 816 of file stl\_map.h.

#### 4.704.3.32 insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
 std::initializer_list< value_type > __list) [inline]
```

Attempts to insert a list of `std::pairs` into the map.

## Parameters

|                     |                                                                                 |
|---------------------|---------------------------------------------------------------------------------|
| <code>__list</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
|---------------------|---------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 830 of file `stl_map.h`.

References `std::map<_Key, _Tp, _Compare, _Alloc >::insert()`.

4.704.3.33 `insert()` [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Attempts to insert a `std::pair` into the map.

## Parameters

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 860 of file `stl_map.h`.

4.704.3.34 `insert()` [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Attempts to insert a `std::pair` into the map.



## Parameters

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 870 of file `stl_map.h`.

References `std::move()`.

## 4.704.3.35 insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, iterator> std::map< _Key, _Tp, _↵
_Compare, _Alloc >::insert (
 const_iterator __position,
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the map.

## Parameters

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 875 of file `stl_map.h`.

#### 4.704.3.36 `insert()` [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Template function that attempts to insert a range of elements.

##### Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 893 of file `stl_map.h`.

#### 4.704.3.37 `key_comp()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the key comparison object out of which the map was constructed.

Definition at line 1142 of file `stl_map.h`.



Referenced by `std::map< _Key, _Tp, _Compare, _Alloc >::at()`.

#### 4.704.3.38 `lower_bound()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 1258 of file stl\_map.h.



Referenced by std::map< \_Key, \_Tp, \_Compare, \_Alloc >::at().

## 4.704.3.39 lower\_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                                                                                     |                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.



Definition at line 1264 of file stl\_map.h.

## 4.704.3.40 lower\_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.



Definition at line 1283 of file stl\_map.h.

**4.704.3.41 lower\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                     |                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 1289 of file stl\_map.h.

**4.704.3.42 max\_size()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the map.

Definition at line 475 of file stl\_map.h.

## 4.704.3.43 operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
 const map<_Key, _Tp, _Compare, _Alloc > &) [default]
```

Map assignment operator.

Whether the allocator is copied depends on the allocator traits.

## 4.704.3.44 operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
 map<_Key, _Tp, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

## 4.704.3.45 operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map& std::map<_Key, _Tp, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Map list assignment operator.

## Parameters

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a map with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

Definition at line 337 of file stl\_map.h.

**4.704.3.46 operator[]()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type& std::map<_Key, _Tp, _Compare, _Alloc >::operator[] (
 const key_type & __k) [inline]
```

Subscript ( [] ) access to map data.

**Parameters**

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

**Returns**

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires logarithmic time.

Definition at line 492 of file stl\_map.h.

**4.704.3.47 rbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 392 of file stl\_map.h.

**4.704.3.48 rbegin()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map<_Key, _Tp, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

Definition at line 401 of file stl\_map.h.

**4.704.3.49** `rend()` [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 410 of file `stl_map.h`.

**4.704.3.50** `rend()` [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

Definition at line 419 of file `stl_map.h`.

**4.704.3.51** `size()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the map.

Definition at line 470 of file `stl_map.h`.

**4.704.3.52** `swap()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
 map< _Key, _Tp, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another map.

**Parameters**

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A map of the same element and allocator types. |
|------------------|------------------------------------------------|

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 1122 of file `stl_map.h`.

#### 4.704.3.53 `upper_bound()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 1303 of file `stl_map.h`.

#### 4.704.3.54 `upper_bound()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) -> decltype(iterator(_M.t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|



**Returns**

Iterator pointing to the first element greater than key, or end().

Definition at line 1309 of file stl\_map.h.

**4.704.3.55 upper\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 1323 of file stl\_map.h.

**4.704.3.56 upper\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 1329 of file `stl_map.h`.

#### 4.704.3.57 `value_comp()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.

Definition at line 1150 of file `stl_map.h`.

The documentation for this class was generated from the following file:

- [stl\\_map.h](#)

### 4.705 `std::mask_array<_Tp>` Class Template Reference

#### Public Types

- typedef `_Tp` **value\_type**

#### Public Member Functions

- `mask_array` (const `mask_array` &)
- void `operator%=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator%=(const _Expr<_Dom, _Tp> &)` const
- void `operator&=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator&=(const _Expr<_Dom, _Tp> &)` const
- void `operator*=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator*=(const _Expr<_Dom, _Tp> &)` const
- void `operator+=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator+=(const _Expr<_Dom, _Tp> &)` const
- void `operator-=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator-=(const _Expr<_Dom, _Tp> &)` const
- void `operator/=(const valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator/=(const _Expr<_Dom, _Tp> &)` const
- void `operator<=<=` (const `valarray<_Tp> &)` const
- template<class `_Dom` >  
void `operator<<=<=` (const `_Expr<_Dom, _Tp> &)` const
- `mask_array` & `operator=` (const `mask_array` &)

- void **operator=** (const [valarray](#)< \_Tp > &) const
- void **operator=** (const \_Tp &) const
- template<class \_Dom >  
void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- template<class \_Ex >  
void **operator=** (const \_Expr< \_Ex, \_Tp > &\_\_e) const
- void **operator>>=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator^=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void **operator|=** (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class [valarray](#)< \_Tp >

#### 4.705.1 Detailed Description

```
template<class _Tp>
class std::mask_array< _Tp >
```

Reference to selected subset of an array.

A `mask_array` is a reference to the actual elements of an array specified by a bitmask in the form of an array of `bool`. The way to get a `mask_array` is to call `operator[]`(`valarray<bool>`) on a `valarray`. The returned `mask_array` then permits carrying operations out on the referenced subset of elements in the original `valarray`.

For example, if a `mask_array` is obtained using the array `(false, true, false, true)` as an argument, the mask array has two elements referring to `array[1]` and `array[3]` in the underlying array.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

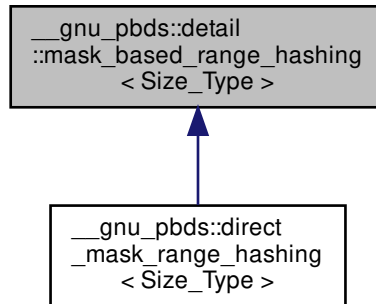
Definition at line 94 of file `valarray`.

The documentation for this class was generated from the following files:

- [valarray](#)
- [mask\\_array.h](#)

#### 4.706 `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mask_based_range_hashing< Size_Type >`:



##### Protected Types

- typedef `Size_Type` **size\_type**

##### Protected Member Functions

- void **notify\_resized** (`size_type` size)
- `size_type` **range\_hash** (`size_type` hash) const
- void **swap** ([mask\\_based\\_range\\_hashing](#) &other)

##### 4.706.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >
```

Range hashing policy.

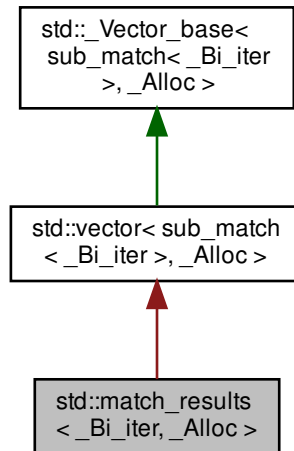
Definition at line 50 of file `mask_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mask\\_based\\_range\\_hashing.hpp](#)

## 4.707 std::match\_results&lt; \_Bi\_iter, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::match\_results< \_Bi\_iter, \_Alloc >:



## Public Types

## 28.10 Public Types

- typedef `sub_match< _Bi_iter > value_type`
- typedef const `value_type` & `const_reference`
- typedef `value_type` & `reference`
- typedef `__Base_type::const_iterator` `const_iterator`
- typedef `const_iterator` `iterator`
- typedef `__iter_traits::difference_type` `difference_type`
- typedef `allocator_traits< _Alloc >::size_type` `size_type`
- typedef `_Alloc` `allocator_type`
- typedef `__iter_traits::value_type` `char_type`
- typedef `std::basic_string< char_type >` `string_type`

## Public Member Functions

- bool `ready` () const noexcept

## 28.10.1 Construction, Copying, and Destruction

- `match_results` ()
- `match_results` (const `_Alloc` &\_\_a) noexcept
- `match_results` (const `match_results` &)=default
- `match_results` (`match_results` &&) noexcept=default

- `match_results & operator=` (`const match_results &`)=default
- `match_results & operator=` (`match_results &&`)=default
- `~match_results` ()=default

### 28.10.2 Size

- `size_type size` () const noexcept
- `size_type max_size` () const noexcept
- `bool empty` () const noexcept

### 28.10.4 Element Access

- `difference_type length` (`size_type __sub=0`) const
- `difference_type position` (`size_type __sub=0`) const
- `string_type str` (`size_type __sub=0`) const
- `const_reference operator[]` (`size_type __sub`) const
- `const_reference prefix` () const
- `const_reference suffix` () const
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator end` () const noexcept
- `const_iterator cend` () const noexcept

### 28.10.5 Formatting

*These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.*

- `template<typename _Out_iter >`  
`_Out_iter format` (`_Out_iter __out`, `const char_type * __fmt_first`, `const char_type * __fmt_last`, `match_flag_type __flags=regex_constants::format_default`) const
- `template<typename _Out_iter, typename _St, typename _Sa >`  
`_Out_iter format` (`_Out_iter __out`, `const basic_string< char_type, _St, _Sa > & __fmt`, `match_flag_type __flags=regex_constants::format_default`) const
- `template<typename _St, typename _Sa >`  
`basic_string< char_type, _St, _Sa > format` (`const basic_string< char_type, _St, _Sa > & __fmt`, `match_flag_type __flags=regex_constants::format_default`) const
- `string_type format` (`const char_type * __fmt`, `match_flag_type __flags=regex_constants::format_default`) const

### 28.10.6 Allocator

- `allocator_type get_allocator` () const noexcept

### 28.10.7 Swap

- `void swap` (`match_results & __that`) noexcept

### Private Types

- `typedef _Alloc_traits::const_pointer` **const\_pointer**
- `typedef std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- `typedef _Base::pointer` **pointer**
- `typedef std::reverse_iterator< iterator >` **reverse\_iterator**

## Private Member Functions

- pointer `_M_allocate` (size\_t \_\_n)
- pointer `_M_allocate` (size\_t \_\_n)
- pointer `_M_allocate_and_copy` (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- void `_M_assign_aux` (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void `_M_assign_aux` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void `_M_assign_dispatch` (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void `_M_assign_dispatch` (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- size\_type `_M_check_len` (size\_type \_\_n, const char \*\_\_s) const
- void `_M_create_storage` (size\_t \_\_n)
- void `_M_deallocate` (pointer \_\_p, size\_t \_\_n)
- void `_M_deallocate` (pointer \_\_p, size\_t \_\_n)
- void `_M_default_append` (size\_type \_\_n)
- void `_M_default_initialize` (size\_type \_\_n)
- auto `_M_emplace_aux` (const\_iterator \_\_position, \_Args &&... \_\_args) -> iterator
- iterator `_M_emplace_aux` (const\_iterator \_\_position, \_Args &&... \_\_args)
- iterator `_M_emplace_aux` (const\_iterator \_\_position, [value\\_type](#) &&\_\_v)
- iterator `_M_erase` (iterator \_\_position)
- iterator `_M_erase` (iterator \_\_first, iterator \_\_last)
- void `_M_erase_at_end` (pointer \_\_pos) noexcept
- void `_M_fill_assign` (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- void `_M_fill_initialize` (size\_type \_\_n, const [value\\_type](#) &\_\_value)
- void `_M_fill_insert` (iterator \_\_pos, size\_type \_\_n, const [value\\_type](#) &\_\_x)
- `_Tp_alloc_type` & `_M_get_Tp_allocator` () noexcept
- const `_Tp_alloc_type` & `_M_get_Tp_allocator` () const noexcept
- `_Tp_alloc_type` & `_M_get_Tp_allocator` () noexcept
- const `_Tp_alloc_type` & `_M_get_Tp_allocator` () const noexcept
- void `_M_insert_aux` (iterator \_\_position, \_Arg &&\_\_arg)
- void `_M_insert_dispatch` (iterator \_\_pos, \_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)
- void `_M_insert_dispatch` (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- iterator `_M_insert_rval` (const\_iterator \_\_position, [value\\_type](#) &&\_\_v)
- void `_M_range_check` (size\_type \_\_n) const
- void `_M_range_initialize` (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void `_M_range_initialize` (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void `_M_range_insert` (iterator \_\_pos, \_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- void `_M_range_insert` (iterator \_\_pos, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- void `_M_realloc_insert` (iterator \_\_position, \_Args &&... \_\_args)
- bool `_M_shrink_to_fit` ()
- void `assign` (size\_type \_\_n, const [value\\_type](#) &\_\_val)
- void `assign` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void `assign` (initializer\_list< [value\\_type](#) > \_\_l)
- `reference at` (size\_type \_\_n)
- `const_reference at` (size\_type \_\_n) const
- `reference back` () noexcept
- `const_reference back` () const noexcept
- iterator `begin` () noexcept
- size\_type `capacity` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator cbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept

- `sub_match<_Bi_iter> * data ()` noexcept
- `const sub_match<_Bi_iter> * data ()` const noexcept
- iterator `emplace (const_iterator __position, _Args &&... __args)`
- void `emplace_back (_Args &&... __args)`
- iterator `end ()` noexcept
- iterator `erase (const_iterator __position)`
- iterator `erase (const_iterator __first, const_iterator __last)`
- `reference front ()` noexcept
- `const_reference front ()` const noexcept
- allocator\_type `get_allocator ()` const noexcept
- iterator `insert (const_iterator __position, const value_type &__x)`
- iterator `insert (const_iterator __position, value_type &&__x)`
- iterator `insert (const_iterator __position, initializer_list<value_type> __l)`
- iterator `insert (const_iterator __position, size_type __n, const value_type &__x)`
- iterator `insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `reference operator[] (size_type __n)` noexcept
- `const_reference operator[] (size_type __n)` const noexcept
- void `pop_back ()` noexcept
- void `push_back (const value_type &__x)`
- void `push_back (value_type &&__x)`
- `reverse_iterator rbegin ()` noexcept
- `const_reverse_iterator rbegin ()` const noexcept
- `reverse_iterator rend ()` noexcept
- `const_reverse_iterator rend ()` const noexcept
- void `reserve (size_type __n)`
- void `resize (size_type __new_size)`
- void `resize (size_type __new_size, const value_type &__x)`
- void `shrink_to_fit ()`
- void `swap (vector &__x)` noexcept

#### Static Private Member Functions

- static `size_type _S_check_init_len (size_type __n, const allocator_type &__a)`
- static `size_type _S_max_size (const _Tp_alloc_type &__a)` noexcept

#### Private Attributes

- `_Vector_impl _M_impl`

#### Friends

- `template<typename , typename , typename >`  
`class regex_iterator`



## 4.707.1 Detailed Description

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results< _Bi_iter, _Alloc >
```

The results of a match or search operation.

A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template match\_results.

This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.

The sub\_match object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the sub\_↵ match member matched is always true. The sub\_match object stored at index n denotes what matched the marked sub-expression n within the matched expression. If the sub-expression n participated in a regular expression match then the sub\_match member matched evaluates to true, and members first and second denote the range of characters [first, second) which formed that match. Otherwise matched is false, and members first and second point to the end of the sequence that was searched.

Definition at line 39 of file regex.h.

## 4.707.2 Constructor &amp; Destructor Documentation

## 4.707.2.1 match\_results() [1/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results () [inline]
```

Constructs a default match\_results container.

**Postcondition**

size() returns 0 and str() returns an empty string.

Definition at line 1728 of file regex.h.

## 4.707.2.2 match\_results() [2/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 const _Alloc & __a) [inline], [explicit], [noexcept]
```

Constructs a default match\_results container.

**Postcondition**

size() returns 0 and str() returns an empty string.

Definition at line 1735 of file regex.h.

#### 4.707.2.3 match\_results() [3/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 const match_results< _Bi_iter, _Alloc > &) [default]
```

Copy constructs a match\_results.

#### 4.707.2.4 match\_results() [4/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
 match_results< _Bi_iter, _Alloc > &&) [default], [noexcept]
```

Move constructs a match\_results.

#### 4.707.2.5 ~match\_results()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::~~match_results () [default]
```

Destroys a match\_results object.

### 4.707.3 Member Function Documentation

#### 4.707.3.1 begin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::begin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

Definition at line 1908 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::cbegin(), and std::operator==( ).

#### 4.707.3.2 cbegin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cbegin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub\_match collection.

Definition at line 1915 of file regex.h.

#### 4.707.3.3 cend()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cend () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

Definition at line 1929 of file regex.h.

#### 4.707.3.4 empty()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::empty () const [inline], [noexcept]
```

Indicates if the match\_results contains no results.

##### Return values

|              |                                        |
|--------------|----------------------------------------|
| <i>true</i>  | The match_results object is empty.     |
| <i>false</i> | The match_results object is not empty. |

Definition at line 1804 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::end(), and std::operator==( ).

#### 4.707.3.5 end()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::end () const [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

Definition at line 1922 of file regex.h.

Referenced by std::match\_results< \_Bi\_iter >::cend(), and std::operator==( ).

#### 4.707.3.6 format() [1/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter >
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
 _Out_iter __out,
 const char_type * __fmt_first,
 const char_type * __fmt_last,
 match_flag_type __flags = regex_constants::format_default) const
```

##### Precondition

ready() == true

Referenced by std::match\_results< \_Bi\_iter >::format().

**4.707.3.7 format()** [2/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter, typename _St, typename _Sa >
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
 _Out_iter __out,
 const basic_string< char_type, _St, _Sa > & __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

ready() == true

Definition at line 1958 of file regex.h.

**4.707.3.8 format()** [3/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _St, typename _Sa >
basic_string<char_type, _St, _Sa> std::match_results< _Bi_iter, _Alloc >::format (
 const basic_string< char_type, _St, _Sa > & __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

ready() == true

Definition at line 1970 of file regex.h.

**4.707.3.9 format()** [4/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::format (
 const char_type * __fmt,
 match_flag_type __flags = regex_constants::format_default) const [inline]
```

**Precondition**

ready() == true

Definition at line 1982 of file regex.h.

#### 4.707.3.10 get\_allocator()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
allocator_type std::match_results<_Bi_iter, _Alloc >::get_allocator () const [inline], [noexcept]
```

Gets a copy of the allocator.

Definition at line 2004 of file regex.h.

#### 4.707.3.11 length()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results<_Bi_iter, _Alloc >::length (
 size_type __sub = 0) const [inline]
```

Gets the length of the indicated submatch.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

##### Precondition

`ready() == true`

This function returns the length of the indicated submatch, or the length of the entire match if `__sub` is zero (the default).

Definition at line 1823 of file regex.h.

#### 4.707.3.12 max\_size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results<_Bi_iter, _Alloc >::max_size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count() + 1` if a match was successful. Some matches may be empty.

##### Returns

the number of matches found.

Definition at line 1795 of file regex.h.

**4.707.3.13 operator=()** [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results& std::match_results< _Bi_iter, _Alloc >::operator= (
 const match_results< _Bi_iter, _Alloc > &) [default]
```

Assigns rhs to \*this.

**4.707.3.14 operator=()** [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results& std::match_results< _Bi_iter, _Alloc >::operator= (
 match_results< _Bi_iter, _Alloc > &&) [default]
```

Move-assigns rhs to \*this.

**4.707.3.15 operator[]()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (
 size_type __sub) const [inline]
```

Gets a sub\_match reference for the match or submatch.

**Parameters**

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

**Precondition**

ready() == true

This function gets a reference to the indicated submatch, or the entire match if \_\_sub is zero.

If \_\_sub >= size() then this function returns a sub\_match with a special value indicating no submatch.

Definition at line 1866 of file regex.h.

**4.707.3.16 position()**

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
 size_type __sub = 0) const [inline]
```

Gets the offset of the beginning of the indicated submatch.

## Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

## Precondition

`ready() == true`

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of `__sub` is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

Definition at line 1838 of file `regex.h`.

4.707.3.17 `prefix()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::prefix () const [inline]
```

Gets a `sub_match` representing the match prefix.

## Precondition

`ready() == true`

This function gets a reference to a `sub_match` object representing the part of the target range between the start of the target range and the start of the match.

Definition at line 1883 of file `regex.h`.

Referenced by `std::operator==()`.

4.707.3.18 `ready()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::ready () const [inline], [noexcept]
```

Indicates if the `match_results` is ready.

## Return values

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>true</i>  | The object has a fully-established result state. |
| <i>false</i> | The object is not ready.                         |

Definition at line 1774 of file regex.h.

Referenced by `std::operator==()`.

#### 4.707.3.19 `size()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::size () const [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or `mark_count()` + 1 if a match was successful. Some matches may be empty.

##### Returns

the number of matches found.

Definition at line 1791 of file regex.h.

Referenced by `std::match_results< _Bi_iter >::empty()`, and `std::operator==()`.

#### 4.707.3.20 `str()`

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::str (
 size_type __sub = 0) const [inline]
```

Gets the match or submatch converted to a string type.

##### Parameters

|                    |                         |
|--------------------|-------------------------|
| <code>__sub</code> | indicates the submatch. |
|--------------------|-------------------------|

##### Precondition

`ready() == true`

This function gets the submatch (or match, if `__sub` is zero) extracted from the target range and converted to the associated string type.

Definition at line 1851 of file regex.h.



## 4.707.3.21 suffix()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::suffix () const [inline]
```

Gets a sub\_match representing the match suffix.

**Precondition**

ready() == true

This function gets a reference to a sub\_match object representing the part of the target range between the end of the match and the end of the target range.

Definition at line 1898 of file regex.h.

Referenced by std::operator==( ).

## 4.707.3.22 swap()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
void std::match_results< _Bi_iter, _Alloc >::swap (
 match_results< _Bi_iter, _Alloc > & __that) [inline], [noexcept]
```

Swaps the contents of two match\_results.

Definition at line 2018 of file regex.h.

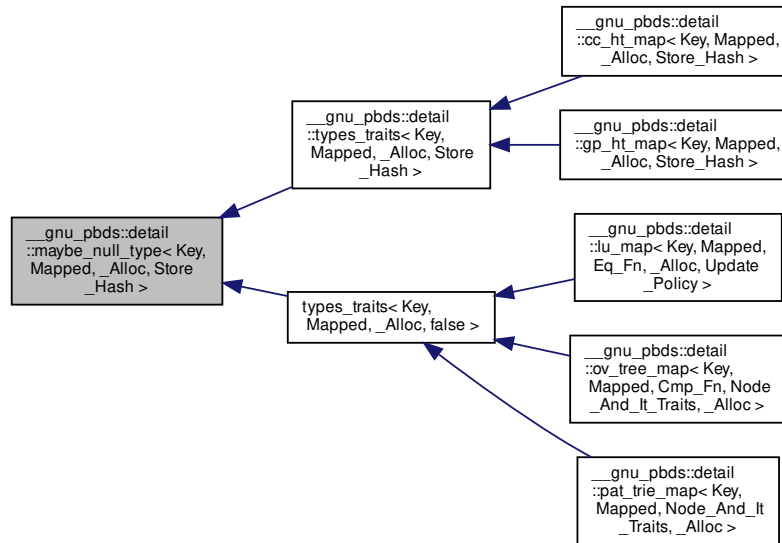
Referenced by std::match\_results< \_Bi\_iter >::swap( ).

The documentation for this class was generated from the following file:

- [regex.h](#)

#### 4.708 `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >`:



##### 4.708.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
```

Base class for conditionally defining a static data member.

Definition at line 121 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.709 `__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >` Struct Template Reference

##### Static Public Attributes

- static `null_type` `s_null_type`

## 4.709.1 Detailed Description

```
template<typename Key, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
```

Specialization that defines a static data member of type null\_type.

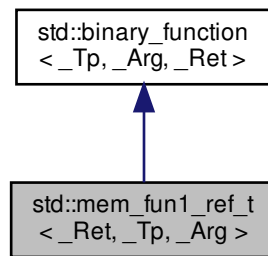
Definition at line 126 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.710 std::mem\_fun1\_ref\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >:



## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Ret [result\\_type](#)
- typedef \_Arg [second\\_argument\\_type](#)

## Public Member Functions

- **mem\_fun1\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)(\_Arg))
- \_Ret **operator()** (\_Tp &\_\_r, \_Arg \_\_x) const

#### 4.710.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1311 of file `stl_function.h`.

#### 4.710.2 Member Typedef Documentation

##### 4.710.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Arg , _Ret >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.710.2.2 result\_type

```
typedef _Ret std::binary_function< _Tp , _Arg , _Ret >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.710.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp , _Arg , _Ret >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

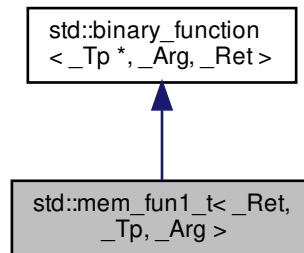
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.711 std::mem\_fun1\_t&lt; \_Ret, \_Tp, \_Arg &gt; Class Template Reference

Inheritance diagram for std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >:



#### Public Types

- typedef `_Tp *` [first\\_argument\\_type](#)
- typedef `_Ret` [result\\_type](#)
- typedef `_Arg` [second\\_argument\\_type](#)

#### Public Member Functions

- **mem\_fun1\_t** (`_Ret`(`_Tp::*__pf`)(`_Arg`))
- `_Ret` **operator()** (`_Tp *``__p`, `_Arg` `__x`) const

#### 4.711.1 Detailed Description

```
template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t< _Ret, _Tp, _Arg >
```

One of the [adaptors for member pointers](#).

Definition at line 1275 of file `stl_function.h`.

#### 4.711.2 Member Typedef Documentation

#### 4.711.2.1 first\_argument\_type

```
typedef _Tp * std::binary_function< _Tp * , _Arg , _Ret >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.711.2.2 result\_type

```
typedef _Ret std::binary_function< _Tp * , _Arg , _Ret >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.711.2.3 second\_argument\_type

```
typedef _Arg std::binary_function< _Tp * , _Arg , _Ret >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

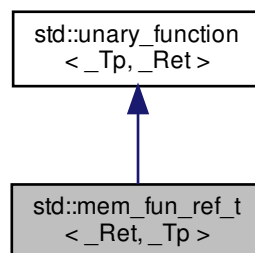
Definition at line 124 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 4.712 std::mem\_fun\_ref\_t<\_Ret, \_Tp> Class Template Reference

Inheritance diagram for std::mem\_fun\_ref\_t<\_Ret, \_Tp>:



### Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Ret [result\\_type](#)

### Public Member Functions

- **mem\_fun\_ref\_t** (\_Ret(\_Tp::\*\_\_pf)())
- **\_Ret operator()** (\_Tp &\_\_r) const

#### 4.712.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::mem_fun_ref_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1239 of file stl\_function.h.

#### 4.712.2 Member Typedef Documentation

##### 4.712.2.1 argument\_type

```
typedef _Tp std::unary_function<_Tp, _Ret>::argument_type [inherited]
```

[argument\\_type](#) is the type of the argument

Definition at line 108 of file stl\_function.h.

##### 4.712.2.2 result\_type

```
typedef _Ret std::unary_function<_Tp, _Ret>::result_type [inherited]
```

[result\\_type](#) is the return type

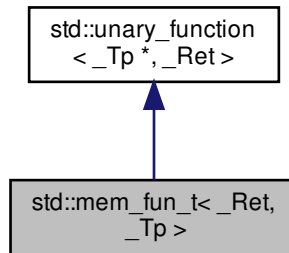
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.713 `std::mem_fun_t<_Ret, _Tp>` Class Template Reference

Inheritance diagram for `std::mem_fun_t<_Ret, _Tp>`:



##### Public Types

- typedef `_Tp *` [argument\\_type](#)
- typedef `_Ret` [result\\_type](#)

##### Public Member Functions

- **`mem_fun_t`** (`_Ret(_Tp::*__pf)()`)
- `_Ret operator()` (`_Tp *__p`) const

##### 4.713.1 Detailed Description

```
template<typename _Ret, typename _Tp>
class std::mem_fun_t<_Ret, _Tp>
```

One of the [adaptors for member pointers](#).

Definition at line 1203 of file `stl_function.h`.

##### 4.713.2 Member Typedef Documentation



#### 4.713.2.1 `argument_type`

```
typedef _Tp * std::unary_function< _Tp * , _Ret >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.713.2.2 `result_type`

```
typedef _Ret std::unary_function< _Tp * , _Ret >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 4.714 `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>` Class Template Reference

#### Public Types

- typedef `_UIntType` [result\\_type](#)

#### Public Member Functions

- `mersenne_twister_engine` ([result\\_type](#) \_\_sd)
- template<typename `_Sseq` , typename `=_If_seed_seq<_Sseq>>`  
`mersenne_twister_engine` (`_Sseq` &\_\_q)
- void `discard` (unsigned long long \_\_z)
- [result\\_type](#) `operator()` ()
- template<typename `_Sseq` >  
auto `seed` (`_Sseq` &\_\_q) -> `_If_seed_seq<_Sseq>`
- void `seed` ([result\\_type](#) \_\_sd=default\_seed)
- template<typename `_Sseq` >  
`_If_seed_seq<_Sseq>` `seed` (`_Sseq` &\_\_q)

#### Static Public Member Functions

- static constexpr [result\\_type](#) `max` ()
- static constexpr [result\\_type](#) `min` ()

## Static Public Attributes

- static constexpr [result\\_type](#) **default\_seed**
- static constexpr [result\\_type](#) **initialization\_multiplier**
- static constexpr size\_t **mask\_bits**
- static constexpr size\_t **shift\_size**
- static constexpr size\_t **state\_size**
- static constexpr [result\\_type](#) **tempering\_b**
- static constexpr [result\\_type](#) **tempering\_c**
- static constexpr [result\\_type](#) **tempering\_d**
- static constexpr size\_t **tempering\_l**
- static constexpr size\_t **tempering\_s**
- static constexpr size\_t **tempering\_t**
- static constexpr size\_t **tempering\_u**
- static constexpr size\_t **word\_size**
- static constexpr [result\\_type](#) **xor\_mask**

## Friends

- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits>  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::mersenne\\_twister\\_engine](#)< \_UIntType1, \_\_w1, \_\_n1, \_\_m1, \_\_r1, \_\_a1, \_\_u1, \_\_d1, \_\_s1, \_\_b1, \_\_t1, \_\_c1, \_\_l1, \_\_f1 > &\_\_x)
- bool [operator==](#) (const [mersenne\\_twister\\_engine](#) &\_\_lhs, const [mersenne\\_twister\\_engine](#) &\_\_rhs)
- template<typename \_UIntType1, size\_t \_\_w1, size\_t \_\_n1, size\_t \_\_m1, size\_t \_\_r1, \_UIntType1 \_\_a1, size\_t \_\_u1, \_UIntType1 \_\_d1, size\_t \_\_s1, \_UIntType1 \_\_b1, size\_t \_\_t1, \_UIntType1 \_\_c1, size\_t \_\_l1, \_UIntType1 \_\_f1, typename \_CharT, typename \_Traits>  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::mersenne\\_twister\\_engine](#)< \_UIntType1, \_\_w1, \_\_n1, \_\_m1, \_\_r1, \_\_a1, \_\_u1, \_\_d1, \_\_s1, \_\_b1, \_\_t1, \_\_c1, \_\_l1, \_\_f1 > &\_\_x)

### 4.714.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s,
 _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
```

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.

The best way to use this generator is with the predefined mt19937 class.

This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

#### Template Parameters

|                        |                                                                    |
|------------------------|--------------------------------------------------------------------|
| <code>↵<br/>__w</code> | Word size, the number of bits in each element of the state vector. |
| <code>↵<br/>__n</code> | The degree of recursion.                                           |
| <code>↵<br/>__m</code> | The period parameter.                                              |
| <code>↵<br/>__r</code> | The separation point bit index.                                    |
| <code>↵<br/>__a</code> | The last row of the twist matrix.                                  |
| <code>↵<br/>__u</code> | The first right-shift tempering matrix parameter.                  |
| <code>↵<br/>__d</code> | The first right-shift tempering matrix mask.                       |
| <code>↵<br/>__s</code> | The first left-shift tempering matrix parameter.                   |
| <code>↵<br/>__b</code> | The first left-shift tempering matrix mask.                        |
| <code>↵<br/>__t</code> | The second left-shift tempering matrix parameter.                  |
| <code>↵<br/>__c</code> | The second left-shift tempering matrix mask.                       |
| <code>↵<br/>__l</code> | The second right-shift tempering matrix parameter.                 |
| <code>↵<br/>__f</code> | Initialization multiplier.                                         |

Definition at line 472 of file random.h.

#### 4.714.2 Member Typedef Documentation

##### 4.714.2.1 `result_type`

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
typedef _UIntType std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>::result_type
```

The type of the generated random value.

Definition at line 507 of file random.h.

### 4.714.3 Constructor & Destructor Documentation

#### 4.714.3.1 mersenne\_twister\_engine()

```
template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a, size_t __u,
 UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f>
template<typename Sseq, typename = _If_seed_seq<Sseq>>
std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __f >::mersenne_twister_engine (
 Sseq & __q) [inline], [explicit]
```

Constructs a mersenne\_twister\_engine random number generator engine seeded from the seed sequence \_\_q.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Definition at line 541 of file random.h.

### 4.714.4 Member Function Documentation

#### 4.714.4.1 discard()

```
template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a,
 size_t __u, UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f>
void std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::discard (
 unsigned long long __z)
```

Discard a sequence of random numbers.

Definition at line 432 of file bits/random.tcc.

#### 4.714.4.2 max()

```
template<typename UIntType, size_t __w, size_t __n, size_t __m, size_t __r, UIntType __a, size_t __u,
 UIntType __d, size_t __s, UIntType __b, size_t __t, UIntType __c, size_t __l, UIntType __f>
static constexpr result_type std::mersenne_twister_engine< UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::max () [inline], [static]
```

Gets the largest possible value in the output range.

Definition at line 562 of file random.h.

#### 4.714.4.3 min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min () [inline], [static]
```

Gets the smallest possible value in the output range.

Definition at line 555 of file random.h.

### 4.714.5 Friends And Related Function Documentation

#### 4.714.5.1 operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1 , size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]
```

Inserts the current state of a % mersenne\_twister\_engine random number generator engine \_\_x into the output stream \_\_os.

#### Parameters

|      |                                                             |
|------|-------------------------------------------------------------|
| __os | An output stream.                                           |
| __x  | A % mersenne_twister_engine random number generator engine. |

#### Returns

The output stream with the state of \_\_x inserted or in an error state.

#### 4.714.5.2 operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

```

bool operator== (
 const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __lhs,
 const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > & __rhs) [friend]

```

Compares two % mersenne\_twister\_engine random number generator objects of the same type for equality.

#### Parameters

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| <code>__lhs</code> | A % mersenne_twister_engine random number generator object.       |
| <code>__rhs</code> | Another % mersenne_twister_engine random number generator object. |

#### Returns

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 587 of file random.h.

#### 4.714.5.3 operator>>

```

template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits > & __is,
 std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]

```

Extracts the current state of a % mersenne\_twister\_engine random number generator engine `__x` from the input stream `__is`.

#### Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__is</code> | An input stream.                                            |
| <code>__x</code>  | A % mersenne_twister_engine random number generator engine. |

#### Returns

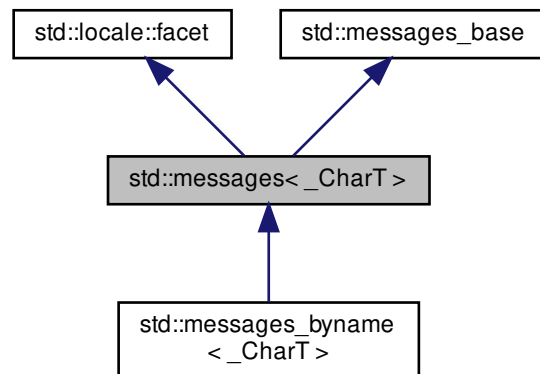
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.715 `std::messages<_CharT>` Class Template Reference

Inheritance diagram for `std::messages<_CharT>`:



#### Public Types

- typedef int **catalog**
- typedef `_CharT` [char\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

#### Public Member Functions

- [messages](#) (size\_t \_\_refs=0)
- [messages](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- [string\\_type](#) **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const [string\\_type](#) &\_\_s) const
- catalog **open** (const [basic\\_string](#)< char > &\_\_s, const [locale](#) &\_\_loc) const
- catalog **open** (const [basic\\_string](#)< char > &, const [locale](#) &, const char \*) const

#### Static Public Attributes

- static [locale::id](#) [id](#)

### Protected Member Functions

- virtual `~messages()`
- `string_type _M_convert_from_char` (char \*) const
- `char * _M_convert_to_char` (const `string_type` & \_\_msg) const
- `template<>`  
`void do_close` (catalog) const
- `template<>`  
`void do_close` (catalog) const
- virtual `void do_close` (catalog) const
- virtual `string_type do_get` (catalog, int, int, const `string_type` & \_\_dfault) const
- `template<>`  
`string do_get` (catalog, int, int, const `string` &) const
- `template<>`  
`wstring do_get` (catalog, int, int, const `wstring` &) const
- `template<>`  
`messages<char>::catalog do_open` (const `basic_string<char>` &, const `locale` &) const
- `template<>`  
`messages<wchar_t>::catalog do_open` (const `basic_string<char>` &, const `locale` &) const
- virtual `catalog do_open` (const `basic_string<char>` &, const `locale` &) const

### Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static `void _S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static `void _S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### Protected Attributes

- `__c_locale _M_c_locale_messages`
- const char \* `_M_name_messages`

#### 4.715.1 Detailed Description

```
template<typename _CharT>
class std::messages< _CharT >
```

Primary class template `messages`.

This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around `gettext`, provided by `libintl`. The second version (ieee) is a wrapper around `catgets`. The final version (default) does no actual translation. These implementations are only provided for `char` and `wchar_t` instantiations.

The `messages` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `messages` facet.

Definition at line 1799 of file `locale_facets_nonio.h`.



#### 4.715.2 Member Typedef Documentation

##### 4.715.2.1 char\_type

```
template<typename _CharT >
typedef _CharT std::messages< _CharT >::char_type
```

Public typedefs.

Definition at line 1805 of file locale\_facets\_nonio.h.

##### 4.715.2.2 string\_type

```
template<typename _CharT >
typedef basic_string<_CharT> std::messages< _CharT >::string_type
```

Public typedefs.

Definition at line 1806 of file locale\_facets\_nonio.h.

#### 4.715.3 Constructor & Destructor Documentation

##### 4.715.3.1 messages() [1/2]

```
template<typename _CharT >
std::messages< _CharT >::messages (
 size_t __refs = 0) [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 44 of file messages\_members.h.

#### 4.715.3.2 messages() [2/2]

```
template<typename _CharT >
std::messages< _CharT >::messages (
 __c_locale __cloc,
 const char * __s,
 size_t __refs = 0) [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

##### Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__s</code>    | The name of a locale.               |
| <code>__refs</code> | Refcount to pass to the base class. |

Definition at line 50 of file messages\_members.h.

#### 4.715.3.3 ~messages()

```
template<typename _CharT >
std::messages< _CharT >::~~messages () [protected], [virtual]
```

Destructor.

Definition at line 79 of file messages\_members.h.

### 4.715.4 Member Function Documentation

#### 4.715.4.1 do\_get()

```
template<>
string std::messages< char >::do_get (
 catalog ,
 int ,
 int ,
 const string &) const [protected]
```

Specializations for required instantiations.

#### 4.715.5 Member Data Documentation

## 4.715.5.1 id

```
template<typename _CharT >
locale::id std::messages< _CharT >::id [static]
```

Numpunct facet id.

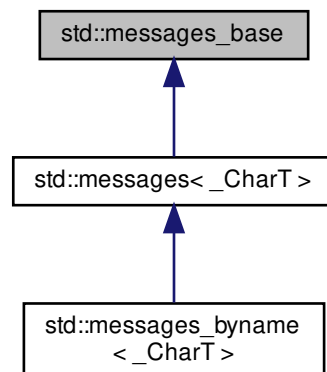
Definition at line 1817 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

## 4.716 std::messages\_base Struct Reference

Inheritance diagram for std::messages\_base:



## Public Types

- typedef int **catalog**

## 4.716.1 Detailed Description

Messages facet base class providing catalog typedef.

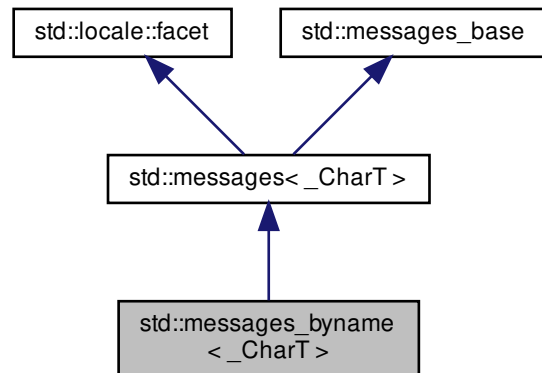
Definition at line 1770 of file locale\_facets\_nonio.h.

The documentation for this struct was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.717 `std::messages_byname<_CharT>` Class Template Reference

Inheritance diagram for `std::messages_byname<_CharT>`:



##### Public Types

- typedef int **catalog**
- typedef `_CharT` **char\_type**
- typedef `basic_string<_CharT>` **string\_type**

##### Public Member Functions

- **messages\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **messages\_byname** (const `string` &\_\_s, size\_t \_\_refs=0)
- void **close** (catalog \_\_c) const
- `string_type` **get** (catalog \_\_c, int \_\_set, int \_\_msgid, const `string_type` &\_\_s) const
- catalog **open** (const `basic_string`< char > &\_\_s, const `locale` &\_\_loc) const
- catalog **open** (const `basic_string`< char > &, const `locale` &, const char \*) const

##### Static Public Attributes

- static `locale::id` **id**

## Protected Member Functions

- [string\\_type](#) **\_M\_convert\_from\_char** (char \*) const
- char \* **\_M\_convert\_to\_char** (const [string\\_type](#) &\_\_msg) const
- template<>  
void **do\_close** (catalog) const
- template<>  
void **do\_close** (catalog) const
- virtual void **do\_close** (catalog) const
- virtual [string\\_type](#) **do\_get** (catalog, int, int, const [string\\_type](#) &\_\_dfault) const
- template<>  
[string](#) **do\_get** (catalog, int, int, const [string](#) &) const
- template<>  
[wstring](#) **do\_get** (catalog, int, int, const [wstring](#) &) const
- template<>  
[messages](#)< char >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- template<>  
[messages](#)< wchar\_t >::catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const
- virtual catalog **do\_open** (const [basic\\_string](#)< char > &, const [locale](#) &) const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- \_\_c\_locale **\_M\_c\_locale\_messages**
- const char \* **\_M\_name\_messages**

## 4.717.1 Detailed Description

```
template<typename _CharT>
class std::messages_byname<_CharT>
```

class messages\_byname [22.2.7.2].

Definition at line 1983 of file locale\_facets\_nonio.h.

## 4.717.2 Member Function Documentation

#### 4.717.2.1 do\_get()

```
template<>
string std::messages< char >::do_get (
 catalog ,
 int ,
 int ,
 const string &) const [protected], [inherited]
```

Specializations for required instantiations.

#### 4.717.3 Member Data Documentation

##### 4.717.3.1 id

```
template<typename _CharT >
locale::id std::messages< _CharT >::id [static], [inherited]
```

Numpunct facet id.

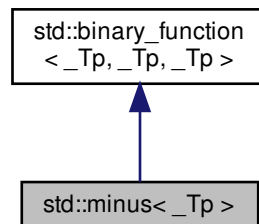
Definition at line 1817 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [messages\\_members.h](#)

#### 4.718 std::minus< \_Tp > Struct Template Reference

Inheritance diagram for std::minus< \_Tp >:



## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef \_Tp [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

## Public Member Functions

- constexpr \_Tp **operator()** (const \_Tp &\_\_x, const \_Tp &\_\_y) const

### 4.718.1 Detailed Description

```
template<typename _Tp>
struct std::minus<_Tp>
```

One of the [math functors](#).

Definition at line 150 of file stl\_function.h.

### 4.718.2 Member Typedef Documentation

#### 4.718.2.1 first\_argument\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::first_argument_type [inherited]
```

[first\\_argument\\_type](#) is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.718.2.2 result\_type

```
typedef _Tp std::binary_function<_Tp, _Tp, _Tp>::result_type [inherited]
```

[result\\_type](#) is the return type

Definition at line 127 of file stl\_function.h.

#### 4.718.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.719 `std::minus< void >` Struct Template Reference

#### Public Types

- typedef `__is_transparent` **is\_transparent**

#### Public Member Functions

- template<typename `_Tp`, typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept([std::forward](#)< `_Tp` >(`_t`) - [std::forward](#)< `_Up` >(`_u`))) -> decltype([std::forward](#)< `_Tp` >(`_t`) - [std::forward](#)< `_Up` >(`_u`))

#### 4.719.1 Detailed Description

```
template<>
struct std::minus< void >
```

One of the [math functors](#).

Definition at line 245 of file `stl_function.h`.

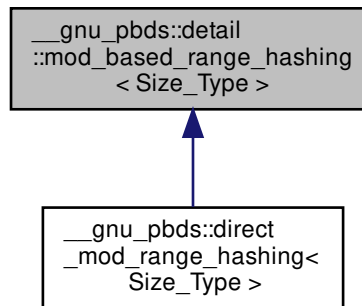
The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)



4.720 `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::mod_based_range_hashing< Size_Type >`:



## Protected Types

- typedef `Size_Type` **size\_type**

## Protected Member Functions

- void **notify\_resized** (size\_type s)
- size\_type **range\_hash** (size\_type s) const
- void **swap** ([mod\\_based\\_range\\_hashing](#) &other)

## 4.720.1 Detailed Description

```
template<typename Size_Type>
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >
```

Mod based range hashing.

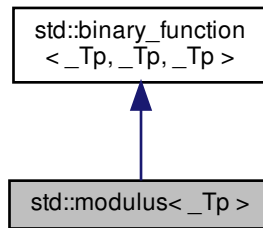
Definition at line 50 of file `mod_based_range_hashing.hpp`.

The documentation for this class was generated from the following file:

- [mod\\_based\\_range\\_hashing.hpp](#)

## 4.721 `std::modulus<_Tp>` Struct Template Reference

Inheritance diagram for `std::modulus<_Tp>`:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 4.721.1 Detailed Description

```
template<typename _Tp>
struct std::modulus<_Tp>
```

One of the [math functors](#).

Definition at line 159 of file `stl_function.h`.

#### 4.721.2 Member Typedef Documentation

## 4.721.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

## 4.721.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

## 4.721.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.722 std::modulus&lt; void &gt; Struct Template Reference

## Public Types

- typedef \_\_is\_transparent **is\_transparent**

## Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) %  
std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) % std::forward< \_Up >(\_\_u))

#### 4.722.1 Detailed Description

```
template<>
struct std::modulus< void >
```

One of the [math functors](#).

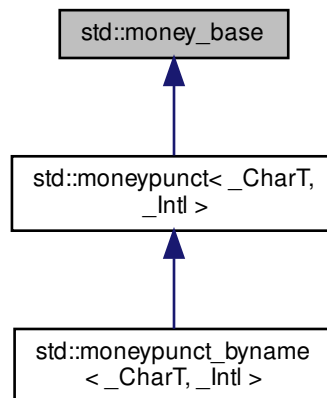
Definition at line 290 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.723 std::money\_base Class Reference

Inheritance diagram for `std::money_base`:



##### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- enum **part** { **none**, **space**, **symbol**, **sign**, **value** }

##### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* `_S_atoms`
- static const pattern `_S_default_pattern`

#### 4.723.1 Detailed Description

Money format ordering data.

This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. `symbol`, `sign`, and `value` must be present and the remaining field must contain either none or space.

### See also

`moneypunct::pos_format()` and `moneypunct::neg_format()` for details of how these fields are interpreted.

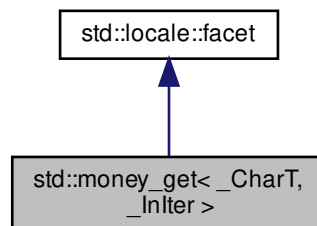
Definition at line 928 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.724 `std::money_get<_CharT, _InIter>` Class Template Reference

Inheritance diagram for `std::money_get<_CharT, _InIter>`:



### Public Types

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)
- typedef [basic\\_string<\\_CharT>](#) [string\\_type](#)

## Public Member Functions

- `money_get` (size\_t \_\_refs=0)
- `template<bool _Intl>`  
`_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11 _Intl _M_extract` (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string & \_\_units) const
- `iter_type get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long double & \_\_units) const
- `iter_type get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string\_type & \_\_digits) const

## Static Public Attributes

- static `locale::id` id

## Protected Member Functions

- virtual `~money_get` ()
- `template<bool _Intl>`  
`iter_type _M_extract` (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string & \_\_digits) const
- virtual `iter_type do_get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, long double & \_\_units) const
- virtual `iter_type do_get` (iter\_type \_\_s, iter\_type \_\_end, bool \_\_intl, ios\_base & \_\_io, ios\_base::iostate & \_\_err, string\_type & \_\_digits) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static `__c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `__c_locale _S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### 4.724.1 Detailed Description

```
template<typename _CharT, typename _Intl>
class std::money_get< _CharT, _Intl >
```

Primary class template `money_get`.

This facet encapsulates the code to parse and return a monetary amount from a string.

The `money_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_get` facet.

Definition at line 1468 of file `locale_facets_nonio.h`.

#### 4.724.2 Member Typedef Documentation

##### 4.724.2.1 char\_type

```
template<typename _CharT , typename _InIter >
typedef _CharT std::money_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 1474 of file locale\_facets\_nonio.h.

##### 4.724.2.2 iter\_type

```
template<typename _CharT , typename _InIter >
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 1475 of file locale\_facets\_nonio.h.

##### 4.724.2.3 string\_type

```
template<typename _CharT , typename _InIter >
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
```

Public typedefs.

Definition at line 1476 of file locale\_facets\_nonio.h.

#### 4.724.3 Constructor & Destructor Documentation

##### 4.724.3.1 money\_get()

```
template<typename _CharT , typename _InIter >
std::money_get< _CharT, _InIter >::money_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

**Parameters**

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1490 of file `locale_facets_nonio.h`.

**4.724.3.2 ~money\_get()**

```
template<typename _CharT , typename _InIter >
virtual std::money_get< _CharT, _InIter >::~~money_get () [inline], [protected], [virtual]
```

Destructor.

Definition at line 1558 of file `locale_facets_nonio.h`.

**4.724.4 Member Function Documentation****4.724.4.1 do\_get()** [1/2]

```
template<typename _CharT , typename _InIter >
_InIter std::money_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __units) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for details.

Definition at line 371 of file `locale_facets_nonio.tcc`.

References `std::basic_string< _CharT, _Traits, _Alloc >::c_str()`.

Referenced by `std::money_get< _CharT, _InIter >::get()`.



## 4.724.4.2 do\_get() [2/2]

```
template<typename _CharT , typename _InIter >
_InIter std::money_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 string_type & __digits) const [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

## See also

get() for details.

Definition at line 384 of file locale\_facets\_nonio.tcc.

References `std::ios_base::M_getloc()`, `std::basic_string< _CharT, _Traits, _Alloc >::resize()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

## 4.724.4.3 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::money_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __units) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in `units` as an integral value `moneypunct::frac_digits()` \* the actual amount. For example, the string \$10.01 in a US locale would store 1001 in `units`.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`. `units` is unchanged if parsing fails.

This function works by returning the result of `do_get()`.

**Parameters**

|                      |                                                                             |
|----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>     | Start of characters to parse.                                               |
| <code>__end</code>   | End of characters to parse.                                                 |
| <code>__intl</code>  | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>    | Source of facets and io state.                                              |
| <code>__err</code>   | Error field to set if parsing fails.                                        |
| <code>__units</code> | Place to store result of parsing.                                           |

**Returns**

Iterator referencing first character beyond valid money amount.

Definition at line 1520 of file `locale_facets_nonio.h`.

References `std::money_get<_CharT, _InIter >::do_get()`.

**4.724.4.4 `get()` [2/2]**

```
template<typename _CharT , typename _InIter >
iter_type std::money_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 bool __intl,
 ios_base & __io,
 ios_base::iostate & __err,
 string_type & __digits) const [inline]
```

Read and parse a monetary value.

This function reads characters from `__s`, interprets them as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and returns the result in *digits*. For example, the string \$10.01 in a US locale would store 1001 in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets `err=(err|io.failbit)`. If the stream is consumed before finishing parsing, sets `err=(err|io.failbit|io.eofbit)`.

This function works by returning the result of `do_get()`.

**Parameters**

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| <code>__s</code>      | Start of characters to parse.                                               |
| <code>__end</code>    | End of characters to parse.                                                 |
| <code>__intl</code>   | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>     | Source of facets and io state.                                              |
| <code>__err</code>    | Error field to set if parsing fails.                                        |
| <code>__digits</code> | Place to store result of parsing.                                           |

### Returns

Iterator referencing first character beyond valid money amount.

Definition at line 1551 of file locale\_facets\_nonio.h.

References `std::money_get< _CharT, _InIter >::do_get()`.

## 4.724.5 Member Data Documentation

### 4.724.5.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::money_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

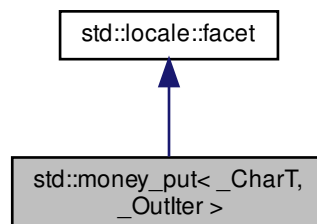
Definition at line 1480 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 4.725 std::money\_put< \_CharT, \_OutIter > Class Template Reference

Inheritance diagram for `std::money_put< _CharT, _OutIter >`:



## Public Types

- typedef `_CharT` `char_type`
- typedef `_Outlter` `iter_type`
- typedef `basic_string<_CharT>` `string_type`

## Public Member Functions

- `money_put` (`size_t` `__refs=0`)
- `template<bool __Intl>`  
`_Outlter _M_insert` (`iter_type` `__s`, `ios_base` & `__io`, `char_type` `__fill`, `const string_type` & `__digits`) `const`
- `iter_type put` (`iter_type` `__s`, `bool` `__intl`, `ios_base` & `__io`, `char_type` `__fill`, `long double` `__units`) `const`
- `iter_type put` (`iter_type` `__s`, `bool` `__intl`, `ios_base` & `__io`, `char_type` `__fill`, `const string_type` & `__digits`) `const`

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~money_put` ()
- `template<bool __Intl>`  
`iter_type _M_insert` (`iter_type` `__s`, `ios_base` & `__io`, `char_type` `__fill`, `const string_type` & `__digits`) `const`
- virtual `iter_type do_put` (`iter_type` `__s`, `bool` `__intl`, `ios_base` & `__io`, `char_type` `__fill`, `long double` `__units`) `const`
- virtual `iter_type do_put` (`iter_type` `__s`, `bool` `__intl`, `ios_base` & `__io`, `char_type` `__fill`, `const string_type` & `__digits`) `const`

## Static Protected Member Functions

- static `__c_locale` `_S_clone_c_locale` (`__c_locale` & `__cloc`) `throw ()`
- static void `_S_create_c_locale` (`__c_locale` & `__cloc`, `const char *``__s`, `__c_locale` `__old=0`)
- static void `_S_destroy_c_locale` (`__c_locale` & `__cloc`)
- static `__c_locale` `_S_get_c_locale` ()
- static `const char *` `_S_get_c_name` () `throw ()`
- static `__c_locale` `_S_lc_ctype_c_locale` (`__c_locale` `__cloc`, `const char *``__s`)

## 4.725.1 Detailed Description

```
template<typename _CharT, typename _Outlter>
class std::money_put<_CharT, _Outlter>
```

Primary class template `money_put`.

This facet encapsulates the code to format and output a monetary amount.

The `money_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `money_put` facet.

Definition at line 1621 of file `locale_facets_nonio.h`.

#### 4.725.2 Member Typedef Documentation

##### 4.725.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::money_put< _CharT, _OutIter >::char_type
```

Public typedefs.

Definition at line 1626 of file locale\_facets\_nonio.h.

##### 4.725.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

Definition at line 1627 of file locale\_facets\_nonio.h.

##### 4.725.2.3 string\_type

```
template<typename _CharT , typename _OutIter >
typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type
```

Public typedefs.

Definition at line 1628 of file locale\_facets\_nonio.h.

#### 4.725.3 Constructor & Destructor Documentation

##### 4.725.3.1 money\_put()

```
template<typename _CharT , typename _OutIter >
std::money_put< _CharT, _OutIter >::money_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1642 of file `locale_facets_nonio.h`.

4.725.3.2 `~money_put()`

```
template<typename _CharT , typename _OutIter >
virtual std::money_put< _CharT, _OutIter >::~~money_put () [inline], [protected], [virtual]
```

Destructor.

Definition at line 1692 of file `locale_facets_nonio.h`.

## 4.725.4 Member Function Documentation

4.725.4.1 `do_put()` <sup>[1/2]</sup>

```
template<typename _CharT , typename _OutIter >
_OutIter std::money_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 long double __units) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to `moneypunct` and `ctype` facets retrieved from `io.getloc()`, and writes the resulting characters to `__s`. For example, the value 1001 in a US locale would write `$10.01` to `__s`.

This function is a hook for derived classes to change the value returned.

## See also

`put()`.

## Parameters

|                     |                                                                             |
|---------------------|-----------------------------------------------------------------------------|
| <code>__s</code>    | The stream to write to.                                                     |
| <code>__intl</code> | Parameter to use <code>_facet&lt;moneypunct&lt;CharT,intl&gt; &gt;</code> . |
| <code>__io</code>   | Source of facets and io state.                                              |
| <code>__fill</code> | <code>char_type</code> to use for padding.                                  |
| <code>units</code>  | Place to store result of parsing.                                           |

**Returns**

Iterator after writing.

Definition at line 577 of file locale\_facets\_nonio.tcc.

References std::ios\_base::getloc(), and std::\_\_ctype\_abstract\_base<\_CharT >::widen().

Referenced by std::money\_put<\_CharT, \_OutIter >::put().

**4.725.4.2 do\_put()** [2/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::money_put<_CharT, _OutIter >::do_put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 const string_type & __digits) const [protected], [virtual]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the string 1001 in a US locale would write \$10.01 to \_\_s.

This function is a hook for derived classes to change the value returned.

**See also**

put().

**Parameters**

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>__s</code>      | The stream to write to.                          |
| <code>__intl</code>   | Parameter to use_facet<moneypunct<CharT,intl> >. |
| <code>__io</code>     | Source of facets and io state.                   |
| <code>__fill</code>   | char_type to use for padding.                    |
| <code>__digits</code> | Place to store result of parsing.                |

**Returns**

Iterator after writing.

Definition at line 615 of file locale\_facets\_nonio.tcc.

#### 4.725.4.3 put() [1/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::money_put< _CharT, _OutIter >::put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 long double __units) const [inline]
```

Format and output a monetary value.

This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the value 1001 in a US locale would write \$10.01 to \_\_s.

This function works by returning the result of do\_put().

##### Parameters

|         |                                                  |
|---------|--------------------------------------------------|
| __s     | The stream to write to.                          |
| __intl  | Parameter to use_facet<moneypunct<CharT,intl> >. |
| __io    | Source of facets and io state.                   |
| __fill  | char_type to use for padding.                    |
| __units | Place to store result of parsing.                |

##### Returns

Iterator after writing.

Definition at line 1662 of file locale\_facets\_nonio.h.

References std::money\_put< \_CharT, \_OutIter >::do\_put().

#### 4.725.4.4 put() [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::money_put< _CharT, _OutIter >::put (
 iter_type __s,
 bool __intl,
 ios_base & __io,
 char_type __fill,
 const string_type & __digits) const [inline]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to \_\_s. For example, the string 1001 in a US locale would write \$10.01 to \_\_s.

This function works by returning the result of do\_put().



## Parameters

|                       |                                                  |
|-----------------------|--------------------------------------------------|
| <code>__s</code>      | The stream to write to.                          |
| <code>__intl</code>   | Parameter to use_facet<moneypunct<CharT,intl> >. |
| <code>__io</code>     | Source of facets and io state.                   |
| <code>__fill</code>   | char_type to use for padding.                    |
| <code>__digits</code> | Place to store result of parsing.                |

## Returns

Iterator after writing.

Definition at line 1685 of file locale\_facets\_nonio.h.

References std::money\_put< \_CharT, \_OutIter >::do\_put().

## 4.725.5 Member Data Documentation

## 4.725.5.1 id

```
template<typename _CharT , typename _OutIter >
locale::id std::money_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

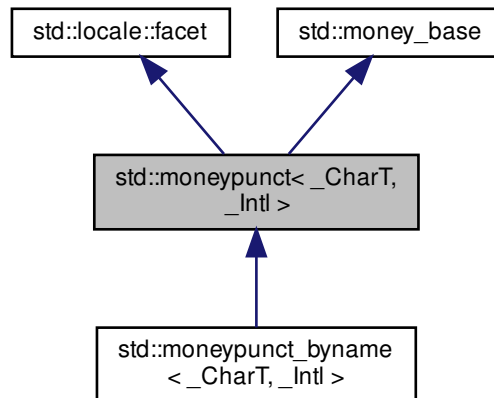
Definition at line 1632 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

#### 4.726 std::moneypunct< \_CharT, \_Intl > Class Template Reference

Inheritance diagram for std::moneypunct< \_CharT, \_Intl >:



##### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > **\_\_cache\_type**
- enum **part** {  
  **none**, **space**, **symbol**, **sign**,  
  **value** }
- typedef \_CharT **char\_type**
- typedef **basic\_string**< \_CharT > **string\_type**

##### Public Member Functions

- **moneypunct** (size\_t \_\_refs=0)
- **moneypunct** (\_\_cache\_type \*\_\_cache, size\_t \_\_refs=0)
- **moneypunct** (\_\_c\_locale \_\_cloc, const char \*\_\_s, size\_t \_\_refs=0)
- **string\_type curr\_symbol** () const
- **char\_type decimal\_point** () const
- **int frac\_digits** () const
- **string grouping** () const
- **string\_type negative\_sign** () const
- **string\_type positive\_sign** () const
- **char\_type thousands\_sep** () const
- pattern **pos\_format** () const
- pattern **neg\_format** () const

## Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

## Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**

## Protected Member Functions

- virtual [~moneypunct](#) ()
- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type](#) **do\_curr\_symbol** () const
- virtual [char\\_type](#) **do\_decimal\_point** () const
- virtual int **do\_frac\_digits** () const
- virtual [string](#) **do\_grouping** () const
- virtual pattern **do\_neg\_format** () const
- virtual [string\\_type](#) **do\_negative\_sign** () const
- virtual pattern **do\_pos\_format** () const
- virtual [string\\_type](#) **do\_positive\_sign** () const
- virtual [char\\_type](#) **do\_thousands\_sep** () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.726.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct<_CharT, _Intl>
```

Primary class template moneypunct.

This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

Definition at line 1024 of file locale\_facets\_nonio.h.

#### 4.726.2 Member Typedef Documentation

##### 4.726.2.1 char\_type

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
```

Public typedefs.

Definition at line 1030 of file locale\_facets\_nonio.h.

##### 4.726.2.2 string\_type

```
template<typename _CharT, bool _Intl>
typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type
```

Public typedefs.

Definition at line 1031 of file locale\_facets\_nonio.h.

#### 4.726.3 Constructor & Destructor Documentation

##### 4.726.3.1 moneypunct() [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1053 of file locale\_facets\_nonio.h.

## 4.726.3.2 moneypunct() [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
 __cache_type * __cache,
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is an internal constructor.

## Parameters

|                      |                                 |
|----------------------|---------------------------------|
| <code>__cache</code> | Cache for optimization.         |
| <code>__refs</code>  | Passed to the base facet class. |

Definition at line 1066 of file locale\_facets\_nonio.h.

## 4.726.3.3 moneypunct() [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
 __c_locale __cloc,
 const char * __s,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__cloc</code> | The C locale.                   |
| <code>__s</code>    | The name of a locale.           |
| <code>__refs</code> | Passed to the base facet class. |

Definition at line 1081 of file locale\_facets\_nonio.h.

## 4.726.3.4 ~moneypunct()

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct< _CharT, _Intl >::~~moneypunct () [protected], [virtual]
```

Destructor.

#### 4.726.4 Member Function Documentation

##### 4.726.4.1 curr\_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline]
```

Return currency symbol string.

This function returns a string\_type to use as a currency symbol. It does so by returning returning moneypunct<char↵\_type>::do\_curr\_symbol().

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1151 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_curr\_symbol().

##### 4.726.4.2 decimal\_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning moneypunct<char↵\_type>::do\_decimal\_point().

##### Returns

*char\_type* representing a decimal point.

Definition at line 1095 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_decimal\_point().

## 4.726.4.3 do\_curr\_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected],
[virtual]
```

Return currency symbol string.

This function returns a [string\\_type](#) to use as a currency symbol. This function is a hook for derived classes to change the value returned.

**See also**

[curr\\_symbol\(\)](#) for details.

**Returns**

*string\_type* representing a currency symbol.

Definition at line 1297 of file [locale\\_facets\\_nonio.h](#).

Referenced by [std::moneypunct<\\_CharT, \\_Intl>::curr\\_symbol\(\)](#).

## 4.726.4.4 do\_decimal\_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a [char\\_type](#) to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1259 of file [locale\\_facets\\_nonio.h](#).

Referenced by [std::moneypunct<\\_CharT, \\_Intl>::decimal\\_point\(\)](#).

#### 4.726.4.5 do\_frac\_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const [inline], [protected],
[virtual]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

##### See also

frac\_digits() for details.

##### Returns

Number of digits in amount fraction.

Definition at line 1337 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::frac\_digits().

#### 4.726.4.6 do\_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const [inline], [protected],
[virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1284 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().



#### 4.726.4.7 do\_neg\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

neg\_format() for details.

##### Returns

Pattern for money values.

Definition at line 1365 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::neg\_format().

#### 4.726.4.8 do\_negative\_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const [inline], [protected],
[virtual]
```

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

##### See also

negative\_sign() for details.

##### Returns

*string\_type* representing a negative sign.

Definition at line 1323 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::negative\_sign().

#### 4.726.4.9 do\_pos\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

pos\_format() for details.

##### Returns

Pattern for money values.

Definition at line 1351 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::pos\_format().

#### 4.726.4.10 do\_positive\_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual]
```

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

##### See also

positive\_sign() for details.

##### Returns

*string\_type* representing a positive sign.

Definition at line 1310 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::positive\_sign().

## 4.726.4.11 do\_thousands\_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct<_CharT, _Intl>::do_thousands_sep () const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a thousands separator.

Definition at line 1271 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct<\_CharT, \_Intl>::thousands\_sep().

## 4.726.4.12 frac\_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct<_CharT, _Intl>::frac_digits () const [inline]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char\_type>::do\_frac\_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac\_digits() digits.

**Returns**

Number of digits in amount fraction.

Definition at line 1201 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_frac\_digits().

#### 4.726.4.13 grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns \003\002 and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char\_type>::do\_grouping().

##### Returns

string representing grouping specification.

Definition at line 1138 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_grouping().

#### 4.726.4.14 neg\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

##### Returns

Pattern for money values.

Definition at line 1241 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_neg\_format().

## 4.726.4.15 negative\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct<_CharT, _Intl>::negative_sign () const [inline]
```

Return negative sign string.

This function returns a string\_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char\_type>::do\_negative\_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg\_format() and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a negative sign.

Definition at line 1185 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_negative\_sign().

## 4.726.4.16 pos\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct<_CharT, _Intl>::pos_format () const [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

**Returns**

Pattern for money values.

Definition at line 1237 of file locale\_facets\_nonio.h.

References std::moneypunct<\_CharT, \_Intl>::do\_pos\_format().

#### 4.726.4.17 positive\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. It does so by returning `returning moneypunct<char_type>::do_positive_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `pos_format()` and the remainder appear at the end of the formatted string.

##### Returns

*string\_type* representing a positive sign.

Definition at line 1168 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_positive_sign()`.

#### 4.726.4.18 thousands\_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a `char_type` to use as a thousands separator. It does so by returning `returning moneypunct<char_type>::do_thousands_sep()`.

##### Returns

`char_type` representing a thousands separator.

Definition at line 1108 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_thousands_sep()`.

#### 4.726.5 Member Data Documentation

## 4.726.5.1 id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id [static]
```

Numpunct facet id.

Definition at line 1043 of file locale\_facets\_nonio.h.

## 4.726.5.2 intl

```
template<typename _CharT, bool _Intl>
const bool std::moneypunct< _CharT, _Intl >::intl [static]
```

This value is provided by the standard, but no reason for its existence.

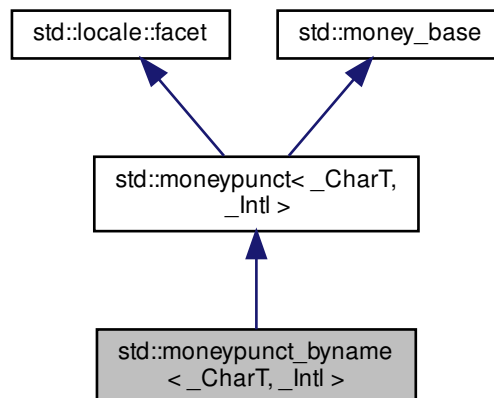
Definition at line 1041 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.727 std::moneypunct\_byname&lt; \_CharT, \_Intl &gt; Class Template Reference

Inheritance diagram for std::moneypunct\_byname< \_CharT, \_Intl >:



### Public Types

- enum { **\_S\_minus**, **\_S\_zero**, **\_S\_end** }
- typedef \_\_moneypunct\_cache< \_CharT, \_Intl > **\_\_cache\_type**
- typedef \_CharT **char\_type**
- enum **part** {  
    **none**, **space**, **symbol**, **sign**,  
    **value** }
- typedef [basic\\_string](#)< \_CharT > **string\_type**

### Public Member Functions

- **moneypunct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
  - **moneypunct\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
  - [string\\_type](#) **curr\_symbol** () const
  - [char\\_type](#) **decimal\_point** () const
  - int **frac\_digits** () const
  - [string](#) **grouping** () const
  - [string\\_type](#) **negative\_sign** () const
  - [string\\_type](#) **positive\_sign** () const
  - [char\\_type](#) **thousands\_sep** () const
- 
- pattern [pos\\_format](#) () const
  - pattern [neg\\_format](#) () const

### Static Public Member Functions

- static pattern **\_S\_construct\_pattern** (char \_\_precedes, char \_\_space, char \_\_posn) throw ()

### Static Public Attributes

- static const char \* **\_S\_atoms**
- static const pattern **\_S\_default\_pattern**
- static [locale::id](#) **id**
- static const bool **intl**



## Protected Member Functions

- void **\_M\_initialize\_moneypunct** (\_\_c\_locale \_\_cloc=0, const char \*\_\_name=0)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- template<>  
void **\_M\_initialize\_moneypunct** (\_\_c\_locale, const char \*)
- virtual [string\\_type do\\_curr\\_symbol](#) () const
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual int [do\\_frac\\_digits](#) () const
- virtual [string do\\_grouping](#) () const
- virtual pattern [do\\_neg\\_format](#) () const
- virtual [string\\_type do\\_negative\\_sign](#) () const
- virtual pattern [do\\_pos\\_format](#) () const
- virtual [string\\_type do\\_positive\\_sign](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.727.1 Detailed Description

```
template<typename _CharT, bool _Intl>
class std::moneypunct_byname<_CharT, _Intl>
```

class moneypunct\_byname [22.2.6.4].

Definition at line 1414 of file locale\_facets\_nonio.h.

## 4.727.2 Member Function Documentation

#### 4.727.2.1 curr\_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const [inline], [inherited]
```

Return currency symbol string.

This function returns a string\_type to use as a currency symbol. It does so by returning returning moneypunct<char↵\_type>::do\_curr\_symbol().

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1151 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_curr\_symbol().

#### 4.727.2.2 decimal\_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning moneypunct<char↵\_type>::do\_decimal\_point().

##### Returns

*char\_type* representing a decimal point.

Definition at line 1095 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_decimal\_point().

#### 4.727.2.3 do\_curr\_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.

This function returns a string\_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

##### See also

curr\_symbol() for details.

##### Returns

*string\_type* representing a currency symbol.

Definition at line 1297 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::curr\_symbol().

## 4.727.2.4 do\_decimal\_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1259 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::decimal_point()`.

## 4.727.2.5 do\_frac\_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

**See also**

`frac_digits()` for details.

**Returns**

Number of digits in amount fraction.

Definition at line 1337 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::frac_digits()`.

#### 4.727.2.6 do\_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1284 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::grouping().

#### 4.727.2.7 do\_neg\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

neg\_format() for details.

##### Returns

Pattern for money values.

Definition at line 1365 of file locale\_facets\_nonio.h.

Referenced by std::moneypunct< \_CharT, \_Intl >::neg\_format().

#### 4.727.2.8 do\_negative\_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

##### See also

`negative_sign()` for details.

##### Returns

*string\_type* representing a negative sign.

Definition at line 1323 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::negative_sign()`.

#### 4.727.2.9 do\_pos\_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

##### See also

`pos_format()` for details.

##### Returns

Pattern for money values.

Definition at line 1351 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::pos_format()`.

#### 4.727.2.10 do\_positive\_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.

This function returns a `string_type` to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

#### See also

`positive_sign()` for details.

#### Returns

*string\_type* representing a positive sign.

Definition at line 1310 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::positive_sign()`.

#### 4.727.2.11 do\_thousands\_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

#### Returns

*char\_type* representing a thousands separator.

Definition at line 1271 of file `locale_facets_nonio.h`.

Referenced by `std::moneypunct< _CharT, _Intl >::thousands_sep()`.

#### 4.727.2.12 frac\_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct<_CharT, _Intl >::frac_digits () const [inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning `returning moneypunct<char_type>::do_frac_digits()`.

The fractional part of a money amount is optional. But if it is present, there must be `frac_digits()` digits.

##### Returns

Number of digits in amount fraction.

Definition at line 1201 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_frac_digits()`.

#### 4.727.2.13 grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct<_CharT, _Intl >::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the `grouping()` returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was 32, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling `moneypunct<char_type>::do_grouping()`.

##### Returns

string representing grouping specification.

Definition at line 1138 of file `locale_facets_nonio.h`.

References `std::moneypunct<_CharT, _Intl >::do_grouping()`.

#### 4.727.2.14 neg\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning `std::moneypunct<char_type>::do_pos_format()` or `std::moneypunct<char_type>::do_neg_format()`.

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of `curr_symbol()` may be present. The sign field indicates that the value of `positive_sign()` or `negative_sign()` must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and `pos_format()` pattern {symbol,sign,value,none}, `curr_symbol() == '$'` `positive_sign() == '+'`, and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

##### Returns

Pattern for money values.

Definition at line 1241 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_neg_format()`.

#### 4.727.2.15 negative\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const [inline], [inherited]
```

Return negative sign string.

This function returns a `string_type` to use as a sign for negative amounts. It does so by returning `std::moneypunct<char_type>::do_negative_sign()`.

If the return value contains more than one character, the first character appears in the position indicated by `neg_format()` and the remainder appear at the end of the formatted string.

##### Returns

*string\_type* representing a negative sign.

Definition at line 1185 of file `locale_facets_nonio.h`.

References `std::moneypunct< _CharT, _Intl >::do_negative_sign()`.



## 4.727.2.16 pos\_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char\_type>::do\_pos\_format() or moneypunct<char\_type>::do\_neg\_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr\_symbol() may be present. The sign field indicates that the value of positive\_sign() or negative\_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos\_format() pattern {symbol,sign,value,none}, curr\_symbol() == '\$' positive\_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is \$+10.01.

**Returns**

Pattern for money values.

Definition at line 1237 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_pos\_format().

## 4.727.2.17 positive\_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const [inline], [inherited]
```

Return positive sign string.

This function returns a string\_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char\_type>::do\_positive\_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos\_format() and the remainder appear at the end of the formatted string.

**Returns**

*string\_type* representing a positive sign.

Definition at line 1168 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_positive\_sign().

#### 4.727.2.18 thousands\_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning returning moneypunct<char↵\_type>::do\_thousands\_sep().

##### Returns

char\_type representing a thousands separator.

Definition at line 1108 of file locale\_facets\_nonio.h.

References std::moneypunct< \_CharT, \_Intl >::do\_thousands\_sep().

#### 4.727.3 Member Data Documentation

##### 4.727.3.1 id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 1043 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.728 std::move\_iterator< \_Iterator > Class Template Reference

##### Public Types

- typedef \_\_traits\_type::difference\_type **difference\_type**
- typedef \_\_traits\_type::iterator\_category **iterator\_category**
- using **iterator\_type** = \_Iterator
- typedef \_Iterator **pointer**
- typedef [conditional](#)< [is\\_reference](#)< \_\_base\_ref >::value, typename [remove\\_reference](#)< \_\_base\_ref >::type &&, \_\_base\_ref >::type **reference**
- typedef \_\_traits\_type::value\_type **value\_type**

## Public Member Functions

- constexpr **move\_iterator** (iterator\_type \_\_i)
- template<typename \_Iter >  
constexpr **move\_iterator** (const [move\\_iterator](#)< \_Iter > &\_\_i)
- constexpr iterator\_type **base** () const
- constexpr reference **operator\*** () const
- constexpr [move\\_iterator](#) **operator+** (difference\_type \_\_n) const
- constexpr [move\\_iterator](#) & **operator++** ()
- constexpr [move\\_iterator](#) **operator++** (int)
- constexpr [move\\_iterator](#) & **operator+=** (difference\_type \_\_n)
- constexpr [move\\_iterator](#) **operator-** (difference\_type \_\_n) const
- constexpr [move\\_iterator](#) & **operator--** ()
- constexpr [move\\_iterator](#) **operator--** (int)
- constexpr [move\\_iterator](#) & **operator-=** (difference\_type \_\_n)
- constexpr pointer **operator->** () const
- template<typename \_Iter >  
constexpr [move\\_iterator](#) & **operator=** (const [move\\_iterator](#)< \_Iter > &\_\_i)
- constexpr reference **operator[]** (difference\_type \_\_n) const

## 4.728.1 Detailed Description

```
template<typename _Iterator>
class std::move_iterator< _Iterator >
```

Class template `move_iterator` is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.

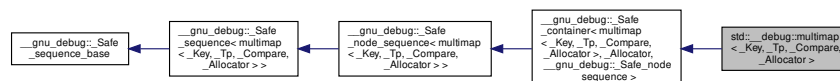
Definition at line 1283 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

4.729 `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_const_iterator, multimap >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::_Safe_iterator<_Base_iterator, multimap >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- **multimap** (const `multimap` &)=default
- **multimap** (`multimap` &&)=default
- **multimap** (`initializer_list< value_type > __l`, const `_Compare` &\_\_c=`_Compare()`, const `allocator_type` &\_\_a=`allocator_type()`)
- **multimap** (const `allocator_type` &\_\_a)
- **multimap** (const `multimap` &\_\_m, const `allocator_type` &\_\_a)
- **multimap** (`multimap` &&\_\_m, const `allocator_type` &\_\_a) noexcept(`noexcept(_Base(std::move(__m._M_base()), __a))`)
- **multimap** (`initializer_list< value_type > __l`, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
**multimap** (`_InputIterator __first`, `_InputIterator __last`, const `allocator_type` &\_\_a)
- **multimap** (const `_Compare` &\_\_comp, const `_Allocator` &\_\_a=`_Allocator()`)
- `template<typename _InputIterator >`  
**multimap** (`_InputIterator __first`, `_InputIterator __last`, const `_Compare` &\_\_comp=`_Compare()`, const `_Allocator` &\_\_a=`_Allocator()`)
- **multimap** (const `_Base` &\_\_x)
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_swap` (`_Safe_container` &\_\_x) noexcept
- void `_M_transfer_from_if` (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **const\_iterator cbegin** () const noexcept
- **const\_iterator cend** () const noexcept
- void **clear** () noexcept
- **const\_reverse\_iterator crbegin** () const noexcept
- **const\_reverse\_iterator crend** () const noexcept
- `template<typename... _Args>`  
**iterator emplace** (`_Args` &&... \_\_args)

- `template<typename... _Args>`  
`iterator` **emplace\_hint** (`const_iterator` \_\_pos, `_Args` &&... \_\_args)
- `iterator` **end** () noexcept
- `const_iterator` **end** () const noexcept
- `std::pair< iterator, iterator >` **equal\_range** (`const key_type` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< iterator, iterator >` **equal\_range** (`const _Kt` &\_\_x)
- `std::pair< const_iterator, const_iterator >` **equal\_range** (`const key_type` &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`std::pair< const_iterator, const_iterator >` **equal\_range** (`const _Kt` &\_\_x) const
- `iterator` **erase** (`const_iterator` \_\_position)
- `_GLIBCXX_ABI_TAG_CXX11` `iterator` **erase** (`iterator` \_\_position)
- `size_type` **erase** (`const key_type` &\_\_x)
- `iterator` **erase** (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator` **find** (`const key_type` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **find** (`const _Kt` &\_\_x)
- `const_iterator` **find** (`const key_type` &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **find** (`const _Kt` &\_\_x) const
- `iterator` **insert** (`const value_type` &\_\_x)
- `iterator` **insert** (`value_type` &&\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator` **insert** (`_Pair` &&\_\_x)
- `void` **insert** (`std::initializer_list< value_type >` \_\_list)
- `iterator` **insert** (`const_iterator` \_\_position, `const value_type` &\_\_x)
- `iterator` **insert** (`const_iterator` \_\_position, `value_type` &&\_\_x)
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator` **insert** (`const_iterator` \_\_position, `_Pair` &&\_\_x)
- `template<typename _InputIterator >`  
`void` **insert** (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `iterator` **lower\_bound** (`const key_type` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **lower\_bound** (`const _Kt` &\_\_x)
- `const_iterator` **lower\_bound** (`const key_type` &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **lower\_bound** (`const _Kt` &\_\_x) const
- `multimap` & **operator=** (`const multimap` &)=default
- `multimap` & **operator=** (`multimap` &&)=default
- `multimap` & **operator=** (`initializer_list< value_type >` \_\_l)
- `reverse_iterator` **rbegin** () noexcept
- `const_reverse_iterator` **rbegin** () const noexcept
- `reverse_iterator` **rend** () noexcept
- `const_reverse_iterator` **rend** () const noexcept
- `void` **swap** (`multimap` &\_\_x) noexcept(*/\*conditional \*/*)
- `iterator` **upper\_bound** (`const key_type` &\_\_x)
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator` **upper\_bound** (`const _Kt` &\_\_x)
- `const_iterator` **upper\_bound** (`const key_type` &\_\_x) const
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator` **upper\_bound** (`const _Kt` &\_\_x) const

### Public Attributes

- `_Safe_iterator_base * \_M\_const\_iterators`
- `_Safe_iterator_base * \_M\_iterators`
- `unsigned int \_M\_version`

### Protected Member Functions

- `void \_M\_detach\_all ()`
- `void \_M\_detach\_singular ()`
- `\_\_gnu\_cxx::\_\_mutex & \_M\_get\_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void \_M\_invalidate\_all () const`
- `void \_M\_revalidate\_singular ()`
- `\_Safe\_container & _M_safe () noexcept`
- `void \_M\_swap (\_Safe\_sequence\_base &__x) noexcept`

### Friends

- `template<typename \_ItT, typename \_SeqT, typename \_CatT >  
class \_\_gnu\_debug::\_Safe\_iterator`

#### 4.729.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::multimap<_Key, _Tp, _Compare, _Allocator >
```

Class `std::multimap` with safety/checking/debug instrumentation.

Definition at line 44 of file `multimap.h`.

#### 4.729.2 Member Function Documentation

##### 4.729.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `\_\_gnu\_debug::\_Safe\_sequence\_base::~~Safe\_sequence\_base\(\)`.

## 4.729.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

**Postcondition**

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

## 4.729.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 4.729.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References \_\_gnu\_debug::\_Safe\_sequence\_base::\_M\_version.

## 4.729.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_invalidate←
_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which \_\_pred(*x*) returns true. \_\_pred will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_sequence.tcc.

#### 4.729.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.729.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.729.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_↵
from_if (
 _Safe_sequence< multimap< _Key, _Tp, _Compare, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 4.729.3 Member Data Documentation

#### 4.729.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.729.3.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.



4.729.3.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [multimap.h](#)

4.730 `std::multimap< _Key, _Tp, _Compare, _Alloc >` Class Template Reference

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Tp` **mapped\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `std::pair< const _Key, _Tp >` **value\_type**

## Public Member Functions

- [multimap](#) ()=default
- [multimap](#) (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- [multimap](#) (const [multimap](#) &)=default
- [multimap](#) ([multimap](#) &&)=default
- [multimap](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- [multimap](#) (const `allocator_type` &\_\_a)
- [multimap](#) (const [multimap](#) &\_\_m, const `allocator_type` &\_\_a)
- [multimap](#) ([multimap](#) &&\_\_m, const `allocator_type` &\_\_a) noexcept([is\\_nothrow\\_copy\\_constructible](#)< `_Compare` >::value && `_Alloc_traits::_S_always_equal`())
- [multimap](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
[multimap](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)

- `template<typename _InputIterator >`  
`multimap` (`_InputIterator __first`, `_InputIterator __last`)
  - `template<typename _InputIterator >`  
`multimap` (`_InputIterator __first`, `_InputIterator __last`, `const _Compare &__comp`, `const allocator_type &__a=allocator_type()`)
  - `~multimap` ()=default
  - `iterator begin` () noexcept
  - `const_iterator begin` () const noexcept
  - `const_iterator cbegin` () const noexcept
  - `const_iterator cend` () const noexcept
  - `void clear` () noexcept
  - `const_reverse_iterator crbegin` () const noexcept
  - `const_reverse_iterator crend` () const noexcept
  - `template<typename... _Args>`  
`iterator emplace` (`_Args &&... __args`)
  - `template<typename... _Args>`  
`iterator emplace_hint` (`const_iterator __pos`, `_Args &&... __args`)
  - `bool empty` () const noexcept
  - `iterator end` () noexcept
  - `const_iterator end` () const noexcept
  - `size_type erase` (`const key_type &__x`)
  - `iterator erase` (`const_iterator __first`, `const_iterator __last`)
  - `allocator_type get_allocator` () const noexcept
  - `template<typename _InputIterator >`  
`void insert` (`_InputIterator __first`, `_InputIterator __last`)
  - `void insert` (`initializer_list< value_type > __l`)
  - `key_compare key_comp` () const
  - `size_type max_size` () const noexcept
  - `multimap & operator=` (`const multimap &`)=default
  - `multimap & operator=` (`multimap &&`)=default
  - `multimap & operator=` (`initializer_list< value_type > __l`)
  - `reverse_iterator rbegin` () noexcept
  - `const_reverse_iterator rbegin` () const noexcept
  - `reverse_iterator rend` () noexcept
  - `const_reverse_iterator rend` () const noexcept
  - `size_type size` () const noexcept
  - `void swap` (`multimap &__x`) noexcept(`/*conditional */`)
  - `value_compare value_comp` () const
- 
- `iterator insert` (`const value_type &__x`)
  - `iterator insert` (`value_type &&__x`)
  - `template<typename _Pair >`  
`__enable_if_t< is_constructible< value_type, _Pair >::value, iterator >` `insert` (`_Pair &&__x`)
- 
- `iterator insert` (`const_iterator __position`, `const value_type &__x`)

- iterator [insert](#) (const\_iterator \_\_position, [value\\_type](#) &&\_\_x)
- template<typename \_Pair >  
[\\_\\_enable\\_if\\_t< is\\_constructible< value\\_type, \\_Pair && >::value, iterator >](#) [insert](#) (const\_iterator \_\_position, [\\_Pair](#) &&\_\_x)
- iterator [erase](#) (const\_iterator \_\_position)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11](#) iterator [erase](#) (iterator \_\_position)
- iterator [find](#) (const key\_type &\_\_x)
- template<typename \_Kt >  
[auto](#) [find](#) (const \_Kt &\_\_x) -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- const\_iterator [find](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
[auto](#) [find](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_find\_tr(\_\_x))
- size\_type [count](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
[auto](#) [count](#) (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- iterator [lower\\_bound](#) (const key\_type &\_\_x)
- template<typename \_Kt >  
[auto](#) [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- const\_iterator [lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt >  
[auto](#) [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- iterator [upper\\_bound](#) (const key\_type &\_\_x)

- `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))`
- `const_iterator upper_bound (const key_type &__x) const`
- `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) const -> decltype(pair< const_iterator, const_iterator >(_M_t._M_equal_range_tr(__x)))`

## Friends

- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator< (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)`
- `template<typename _K1, typename _T1, typename _C1, typename _A1 >`  
`bool operator== (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)`

## 4.730.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const
_Key, _Tp>> >>
class std::multimap< _Key, _Tp, _Compare, _Alloc >
```

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

## Template Parameters

|                       |                                                                                     |
|-----------------------|-------------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                                |
| <code>_Tp</code>      | Type of mapped objects.                                                             |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> .        |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;pair&lt;const _Key, _Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multimap<Key, T>` the `key_type` is `Key`, the `mapped_type` is `T`, and the `value_type` is `std::pair<const Key, T>`.

Multimaps support bidirectional iterators.

The private tree data is declared exactly the same way for `map` and `multimap`; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 72 of file `stl_map.h`.

#### 4.730.2 Constructor & Destructor Documentation

##### 4.730.2.1 multimap() [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap () [default]
```

Default constructor creates no elements.

##### 4.730.2.2 multimap() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap<_Key, _Tp, _Compare, _Alloc >::multimap (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a `multimap` with no elements.

##### Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | A comparison object. |
| <code>__a</code>    | An allocator object. |

Definition at line 191 of file `stl_multimap.h`.

##### 4.730.2.3 multimap() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 const multimap< _Key, _Tp, _Compare, _Alloc > &) [default]
```

Multimap copy constructor.

Whether the allocator is copied depends on the allocator traits.

#### 4.730.2.4 multimap() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 multimap< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Multimap move constructor.

The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

#### 4.730.2.5 multimap() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from an `initializer_list`.

##### Parameters

|                     |                                    |
|---------------------|------------------------------------|
| <code>__l</code>    | An <code>initializer_list</code> . |
| <code>__comp</code> | A comparison functor.              |
| <code>__a</code>    | An allocator object.               |

Create a multimap consisting of copies of the elements from the `initializer_list`. This is linear in  $N$  if the list is already sorted, and  $N\log N$  otherwise (where  $N$  is `__l.size()`).

Definition at line 225 of file `stl_multimap.h`.

#### 4.730.2.6 multimap() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 233 of file `stl_multimap.h`.

**4.730.2.7 multimap()** [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 const multimap<_Key, _Tp, _Compare, _Alloc> & __m,
 const allocator_type & __a) [inline]
```

Allocator-extended copy constructor.

Definition at line 237 of file stl\_multimap.h.

**4.730.2.8 multimap()** [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 multimap<_Key, _Tp, _Compare, _Alloc> && __m,
 const allocator_type & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 241 of file stl\_multimap.h.

**4.730.2.9 multimap()** [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 initializer_list<value_type> __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 247 of file stl\_multimap.h.

**4.730.2.10 multimap()** [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::multimap<_Key, _Tp, _Compare, _Alloc>::multimap (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

Definition at line 253 of file stl\_multimap.h.

**4.730.2.11** `multimap()` [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a multimap from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 269 of file `stl_multimap.h`.

**4.730.2.12** `multimap()` [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multimap from a range.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multimap consisting of copies of the elements from [`__first`,`__last`). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(`__first`,`__last`)).

Definition at line 285 of file `stl_multimap.h`.



## 4.730.2.13 ~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::~~multimap () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 4.730.3 Member Function Documentation

## 4.730.3.1 begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 351 of file stl\_multimap.h.

## 4.730.3.2 begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 360 of file stl\_multimap.h.

## 4.730.3.3 cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 424 of file stl\_multimap.h.

#### 4.730.3.4 cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 433 of file `stl_multimap.h`.

#### 4.730.3.5 clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap<_Key, _Tp, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 808 of file `stl_multimap.h`.

#### 4.730.3.6 count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap<_Key, _Tp, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

##### Parameters

|                  |                                          |
|------------------|------------------------------------------|
| <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------|------------------------------------------|

##### Returns

Number of elements with specified key.

Definition at line 885 of file `stl_multimap.h`.

**4.730.3.7 count()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
 const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements with given key.

**Parameters**

|                 |                                          |
|-----------------|------------------------------------------|
| <code>_↵</code> | Key of (key, value) pairs to be located. |
| <code>_X</code> |                                          |

**Returns**

Number of elements with specified key.

Definition at line 891 of file stl\_multimap.h.

**4.730.3.8 crbegin()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 442 of file stl\_multimap.h.

**4.730.3.9 crend()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 451 of file stl\_multimap.h.

**4.730.3.10 emplace()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Build and insert a std::pair into the multimap.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 491 of file `stl_multimap.h`.

**4.730.3.11 `emplace_hint()`**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Builds and inserts a `std::pair` into the multimap.

**Parameters**

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 518 of file `stl_multimap.h`.

**4.730.3.12 empty()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the multimap is empty.

Definition at line 458 of file stl\_multimap.h.

**4.730.3.13 end()** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

Definition at line 369 of file stl\_multimap.h.

**4.730.3.14 end()** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.


Definition at line 378 of file stl\_multimap.h.

**4.730.3.15 equal\_range()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<iterator, iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                                                                                                      |                                          |
|------------------------------------------------------------------------------------------------------|------------------------------------------|
|  <code>__x</code> | Key of (key, value) pairs to be located. |
|------------------------------------------------------------------------------------------------------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1020 of file stl\_multimap.h.

**4.730.3.16 equal\_range()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                                          |
|-------|------------------------------------------|
| $\_x$ | Key of (key, value) pairs to be located. |
|-------|------------------------------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1026 of file stl\_multimap.h.

**4.730.3.17 equal\_range()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair<const_iterator, const_iterator> std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_
range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

## Parameters

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <a href="#"><math>\_x</math></a> | Key of (key, value) pairs to be located. |
|----------------------------------|------------------------------------------|

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1047 of file stl\_multimap.h.

## 4.730.3.18 equal\_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
 const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator>(_M_t._M_
equal_range_tr(__x))) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <a href="#"><math>\_x</math></a> | Key of (key, value) pairs to be located. |
|----------------------------------|------------------------------------------|

## Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

Definition at line 1053 of file stl\_multimap.h.

**4.730.3.19** `erase()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a multimap.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 702 of file `stl_multimap.h`.

**4.730.3.20** `erase()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from a multimap.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 708 of file `stl_multimap.h`.



**4.730.3.21 erase()** [ 3 / 4 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 739 of file stl\_multimap.h.

**4.730.3.22 erase()** [ 4 / 4 ]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [first,last) range of elements from a multimap.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased .  |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 760 of file stl\_multimap.h.

#### 4.730.3.23 find() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a multimap.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

##### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 843 of file stl\_multimap.h.

#### 4.730.3.24 find() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap<_Key, _Tp, _Compare, _Alloc >::find (
 const _Kt & __x) -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a multimap.

##### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

##### Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.



Definition at line 849 of file stl\_multimap.h.

#### 4.730.3.25 find() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a multimap.

##### Parameters

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

##### Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.



Definition at line 867 of file stl\_multimap.h.

#### 4.730.3.26 find() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
 const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x)) [inline]
```

Tries to locate an element in a multimap.

##### Parameters

|                                                                                     |                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

##### Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 873 of file `stl_multimap.h`.

#### 4.730.3.27 `get_allocator()`

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Get a copy of the memory allocation object.

Definition at line 341 of file `stl_multimap.h`.

#### 4.730.3.28 `insert()` [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

##### Parameters

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------|-----------------------------------------------------------------------------------|

##### Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 539 of file `stl_multimap.h`.

Referenced by `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.

#### 4.730.3.29 `insert()` [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                                   |                                                                      |
|-----------------------------------|----------------------------------------------------------------------|
| <code>↔</code><br><code>_X</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|-----------------------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 546 of file stl\_multimap.h.

References std::move().

**4.730.3.30 insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair>::value, iterator> std::multimap<_Key, _Tp, ↔
_Compare, _Alloc >::insert (
 _Pair && __x) [inline]
```

Inserts a std::pair into the multimap.

**Parameters**

|                                   |                                                                      |
|-----------------------------------|----------------------------------------------------------------------|
| <code>↔</code><br><code>_X</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|-----------------------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.

Insertion requires logarithmic time.

Definition at line 551 of file stl\_multimap.h.

**4.730.3.31** `insert()` [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 579 of file `stl_multimap.h`.

**4.730.3.32** `insert()` [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap<_Key, _Tp, _Compare, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 589 of file `stl_multimap.h`.

References `std::move()`.

**4.730.3.33 `insert()`** [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::multimap< _Key, _Tp,
_Compare, _Alloc >::insert (
 const_iterator __position,
 _Pair && __x) [inline]
```

Inserts a `std::pair` into the multimap.

**Parameters**

|                         |                                                                                   |
|-------------------------|-----------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>        | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a `std::map` the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 594 of file `stl_multimap.h`.

**4.730.3.34** `insert()` [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator >
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 613 of file `stl_multimap.h`.

**4.730.3.35** `insert()` [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of `std::pairs` into the multimap.

**Parameters**

|                         |                                                                                 |
|-------------------------|---------------------------------------------------------------------------------|
| ↔<br>_↔<br>↔<br>_↔<br>/ | A <code>std::initializer_list&lt;value_type&gt;</code> of pairs to be inserted. |
|-------------------------|---------------------------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 625 of file `stl_multimap.h`.

References `std::multimap< _Key, _Tp, _Compare, _Alloc >::insert()`.



**4.730.3.36 key\_comp()**

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the key comparison object out of which the multimap was constructed.

Definition at line 817 of file stl\_multimap.h.

**4.730.3.37 lower\_bound()** [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                         |                                         |
|-------------------------|-----------------------------------------|
| <b><code>__x</code></b> | Key of (key, value) pair to be located. |
|-------------------------|-----------------------------------------|

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.



Definition at line 928 of file stl\_multimap.h.

**4.730.3.38 lower\_bound()** [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.



Definition at line 934 of file stl\_multimap.h.

**4.730.3.39 lower\_bound()** [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                                                                                     |                                         |
|-------------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
|  |                                         |

**Returns**

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().


Definition at line 953 of file stl\_multimap.h.

**4.730.3.40 lower\_bound()** [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                                                                                   |                                         |
|-----------------------------------------------------------------------------------|-----------------------------------------|
|  | Key of (key, value) pair to be located. |
| <code>_X</code>                                                                   |                                         |

## Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

Definition at line 959 of file stl\_multimap.h.

## 4.730.3.41 max\_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the multimap.

Definition at line 468 of file stl\_multimap.h.

## 4.730.3.42 operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
 const multimap< _Key, _Tp, _Compare, _Alloc > &) [default]
```

Multimap assignment operator.

Whether the allocator is copied depends on the allocator traits.

## 4.730.3.43 operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
 multimap< _Key, _Tp, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

## 4.730.3.44 operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap& std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Multimap list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↵ | An initializer_list. |
| ↵ |                      |
| ↵ |                      |
| ↵ |                      |
| / |                      |

This function fills a multimap with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

Definition at line 332 of file `stl_multimap.h`.

**4.730.3.45 `rbegin()`** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 387 of file `stl_multimap.h`.

**4.730.3.46 `rbegin()`** [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 396 of file `stl_multimap.h`.

**4.730.3.47 `rend()`** [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 405 of file `stl_multimap.h`.

**4.730.3.48** rend() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

Definition at line 414 of file stl\_multimap.h.

**4.730.3.49** size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the multimap.

Definition at line 463 of file stl\_multimap.h.

**4.730.3.50** swap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
 multimap< _Key, _Tp, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another multimap.

**Parameters**

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multimap of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 797 of file stl\_multimap.h.

**4.730.3.51** `upper_bound()` [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

Definition at line 973 of file `stl_multimap.h`.

**4.730.3.52** `upper_bound()` [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) -> decltype(iterator(_M.t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

**Returns**

Iterator pointing to the first element greater than key, or end().

Definition at line 979 of file `stl_multimap.h`.

**4.730.3.53** `upper_bound()` [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 993 of file stl\_multimap.h.

#### 4.730.3.54 upper\_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt >
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

#### Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__x</code> | Key of (key, value) pair to be located. |
|------------------|-----------------------------------------|

#### Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

Definition at line 999 of file stl\_multimap.h.

#### 4.730.3.55 value\_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp () const [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.

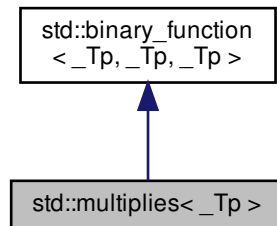
Definition at line 825 of file stl\_multimap.h.

The documentation for this class was generated from the following files:

- [stl\\_map.h](#)
- [stl\\_multimap.h](#)

#### 4.731 `std::multiplies<_Tp>` Struct Template Reference

Inheritance diagram for `std::multiplies<_Tp>`:



##### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

##### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

##### 4.731.1 Detailed Description

```
template<typename _Tp>
struct std::multiplies<_Tp>
```

One of the [math functors](#).

Definition at line 153 of file `stl_function.h`.

##### 4.731.2 Member Typedef Documentation



4.731.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

4.731.2.2 `result_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.731.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

4.732 `std::multiplies< void >` Struct Template Reference

## Public Types

- typedef `__is_transparent` **is\_transparent**

## Public Member Functions

- template<typename `_Tp` , typename `_Up` >  
constexpr auto **operator()** (`_Tp` &&`_t`, `_Up` &&`_u`) const noexcept(noexcept(`std::forward`< `_Tp` >(`_t` ←  
`t`) \*`std::forward`< `_Up` >(`_u`))) -> decltype(`std::forward`< `_Tp` >(`_t`) \*`std::forward`< `_Up` >(`_u`))

#### 4.732.1 Detailed Description

```
template<>
struct std::multiplies< void >
```

One of the [math functors](#).

Definition at line 260 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

#### 4.733 `std::multiset< _Key, _Compare, _Alloc >` Class Template Reference

##### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Rep_type::const_iterator` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `_Rep_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `_Rep_type::difference_type` **difference\_type**
- typedef `_Rep_type::const_iterator` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Alloc_traits::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `_Rep_type::const_reverse_iterator` **reverse\_iterator**
- typedef `_Rep_type::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

##### Public Member Functions

- [multiset](#) ()=default
- [multiset](#) (const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- template<typename `_InputIterator` >  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- template<typename `_InputIterator` >  
[multiset](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp, const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const [multiset](#) &)=default
- [multiset](#) ([multiset](#) &&)=default
- [multiset](#) ([initializer\\_list](#)< `value_type` > \_\_l, const `_Compare` &\_\_comp=\_Compare(), const `allocator_type` &\_\_a=allocator\_type())
- [multiset](#) (const `allocator_type` &\_\_a)
- [multiset](#) (const [multiset](#) &\_\_m, const `allocator_type` &\_\_a)

- `multiset` (`multiset` &&\_\_m, const allocator\_type &\_\_a) noexcept(is\_nothrow\_copy\_constructible< \_Compare > &←  
::value &&\_Alloc\_traits::S\_always\_equal())
  - `multiset` (`initializer_list`< value\_type > \_\_l, const allocator\_type &\_\_a)
  - template<typename \_InputIterator >  
`multiset` (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
  - `~multiset` ()=default
  - iterator `begin` () const noexcept
  - iterator `cbegin` () const noexcept
  - iterator `cend` () const noexcept
  - void `clear` () noexcept
  - `reverse_iterator` `crbegin` () const noexcept
  - `reverse_iterator` `crend` () const noexcept
  - template<typename... \_Args>  
iterator `emplace` (\_Args &&... \_\_args)
  - template<typename... \_Args>  
iterator `emplace_hint` (const\_iterator \_\_pos, \_Args &&... \_\_args)
  - bool `empty` () const noexcept
  - iterator `end` () const noexcept
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_position)
  - size\_type `erase` (const key\_type &\_\_x)
  - \_GLIBCXX\_ABI\_TAG\_CXX11 iterator `erase` (const\_iterator \_\_first, const\_iterator \_\_last)
  - allocator\_type `get_allocator` () const noexcept
  - iterator `insert` (const value\_type &\_\_x)
  - iterator `insert` (value\_type &&\_\_x)
  - iterator `insert` (const\_iterator \_\_position, const value\_type &\_\_x)
  - iterator `insert` (const\_iterator \_\_position, value\_type &&\_\_x)
  - template<typename \_InputIterator >  
void `insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void `insert` (`initializer_list`< value\_type > \_\_l)
  - key\_compare `key_comp` () const
  - size\_type `max_size` () const noexcept
  - `multiset` & `operator=` (const `multiset` &)=default
  - `multiset` & `operator=` (`multiset` &&)=default
  - `multiset` & `operator=` (`initializer_list`< value\_type > \_\_l)
  - `reverse_iterator` `rbegin` () const noexcept
  - `reverse_iterator` `rend` () const noexcept
  - size\_type `size` () const noexcept
  - void `swap` (`multiset` &\_\_x) noexcept(*/\*conditional \*/*)
  - value\_compare `value_comp` () const
- 
- size\_type `count` (const key\_type &\_\_x) const
  - template<typename \_Kt >  
auto `count` (const \_Kt &\_\_x) const -> decltype(\_M\_t.\_M\_count\_tr(\_\_x))
- 
- iterator `find` (const key\_type &\_\_x)

- `const_iterator` [find](#) (const key\_type &\_\_x) const
  - `template<typename _Kt >`  
`auto` [find](#) (const \_Kt &\_\_x) -> decltype(iterator
  - `template<typename _Kt >`  
`auto` [find](#) (const \_Kt &\_\_x) const -> decltype(const\_iterator
- 
- `iterator` [lower\\_bound](#) (const key\_type &\_\_x)
  - `const_iterator` [lower\\_bound](#) (const key\_type &\_\_x) const
  - `template<typename _Kt >`  
`auto` [lower\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
  - `template<typename _Kt >`  
`auto` [lower\\_bound](#) (const \_Kt &\_\_x) const -> decltype(iterator(\_M\_t.\_M\_lower\_bound\_tr(\_\_x)))
- 
- `iterator` [upper\\_bound](#) (const key\_type &\_\_x)
  - `const_iterator` [upper\\_bound](#) (const key\_type &\_\_x) const
  - `template<typename _Kt >`  
`auto` [upper\\_bound](#) (const \_Kt &\_\_x) -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
  - `template<typename _Kt >`  
`auto` [upper\\_bound](#) (const \_Kt &\_\_x) const -> decltype(iterator(\_M\_t.\_M\_upper\_bound\_tr(\_\_x)))
- 
- `std::pair< iterator, iterator >` [equal\\_range](#) (const key\_type &\_\_x)
  - `std::pair< const_iterator, const_iterator >` [equal\\_range](#) (const key\_type &\_\_x) const
  - `template<typename _Kt >`  
`auto` [equal\\_range](#) (const \_Kt &\_\_x) -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))
  - `template<typename _Kt >`  
`auto` [equal\\_range](#) (const \_Kt &\_\_x) const -> decltype(pair< iterator, iterator >(\_M\_t.\_M\_equal\_range\_tr(\_\_x)))

#### Friends

- `template<typename _K1, typename _C1, typename _A1 >`  
`bool` **operator**< (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool` **operator**== (const [multiset](#)< \_K1, \_C1, \_A1 > &, const [multiset](#)< \_K1, \_C1, \_A1 > &)

#### 4.733.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >
```

A standard container made up of elements, which can be retrieved in logarithmic time.

## Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using equivalent keys). For a `multiset<Key>` the `key_type` and `value_type` are `Key`.

Multisets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (`*_unique` versus `*_equal`, same as the standard).

Definition at line 96 of file `stl_multiset.h`.

## 4.733.2 Constructor &amp; Destructor Documentation

## 4.733.2.1 multiset() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset () [default]
```

Default constructor creates no elements.

## 4.733.2.2 multiset() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a multiset with no elements.

## Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

Definition at line 173 of file `stl_multiset.h`.

**4.733.2.3** `multiset()` [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::multiset< _Key, _Compare, _Alloc >::multiset (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a multiset from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a multiset consisting of copies of the elements from `[first,last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 187 of file `stl_multiset.h`.

**4.733.2.4** `multiset()` [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::multiset< _Key, _Compare, _Alloc >::multiset (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from a range.

**Parameters**

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a multiset consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 203 of file `stl_multiset.h`.

## 4.733.2.5 multiset() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc >::multiset (
 const multiset<_Key, _Compare, _Alloc > &) [default]
```

Multiset copy constructor.

Whether the allocator is copied depends on the allocator traits.

## 4.733.2.6 multiset() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc >::multiset (
 multiset<_Key, _Compare, _Alloc > &&) [default]
```

Multiset move constructor.

The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

## 4.733.2.7 multiset() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset<_Key, _Compare, _Alloc >::multiset (
 initializer_list<value_type> & __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a multiset from an initializer\_list.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 239 of file `stl_multiset.h`.

## 4.733.2.8 multiset() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
std::multiset< _Key, _Compare, _Alloc >::multiset (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 247 of file `stl_multiset.h`.

#### 4.733.2.9 multiset() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 const multiset< _Key, _Compare, _Alloc > & __m,
 const allocator_type & __a) [inline]
```

Allocator-extended copy constructor.

Definition at line 251 of file `stl_multiset.h`.

#### 4.733.2.10 multiset() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 multiset< _Key, _Compare, _Alloc > && __m,
 const allocator_type & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 255 of file `stl_multiset.h`.

#### 4.733.2.11 multiset() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 261 of file `stl_multiset.h`.



## 4.733.2.12 multiset() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
template<typename _InputIterator >
std::multiset< _Key, _Compare, _Alloc >::multiset (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

Definition at line 267 of file stl\_multiset.h.

## 4.733.2.13 ~multiset()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
std::multiset< _Key, _Compare, _Alloc >::~~multiset () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 4.733.3 Member Function Documentation

## 4.733.3.1 begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
iterator std::multiset< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 340 of file stl\_multiset.h.

## 4.733.3.2 cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>>
iterator std::multiset< _Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 377 of file stl\_multiset.h.

#### 4.733.3.3 cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 386 of file stl\_multiset.h.

#### 4.733.3.4 clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 718 of file stl\_multiset.h.

#### 4.733.3.5 count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements with given key.

##### Parameters

|                        |                                |
|------------------------|--------------------------------|
| <code>↵<br/>__x</code> | Key of elements to be located. |
|------------------------|--------------------------------|

##### Returns

Number of elements with specified key.

Definition at line 730 of file stl\_multiset.h.

**4.733.3.6 count()** [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset<_Key, _Compare, _Alloc >::count (
 const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements with given key.

**Parameters**

|                  |                                |
|------------------|--------------------------------|
| <code>__x</code> | Key of elements to be located. |
|------------------|--------------------------------|

**Returns**

Number of elements with specified key.

Definition at line 736 of file stl\_multiset.h.

**4.733.3.7 crbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 395 of file stl\_multiset.h.

**4.733.3.8 crend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 404 of file stl\_multiset.h.

**4.733.3.9 emplace()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::multiset<_Key, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

**Parameters**

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate the element instance to be inserted. |
|---------------------|-----------------------------------------------------------------|

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 457 of file `stl_multiset.h`.

**4.733.3.10 `emplace_hint()`**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename... _Args>
iterator std::multiset<_Key, _Compare, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Builds and inserts an element into the multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element instance to be inserted.               |

**Returns**

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 483 of file `stl_multiset.h`.

## 4.733.3.11 empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 410 of file stl\_multiset.h.

## 4.733.3.12 end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

Definition at line 349 of file stl\_multiset.h.

## 4.733.3.13 equal\_range() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<iterator, iterator> std::multiset< _Key, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 879 of file stl\_multiset.h.

**4.733.3.14** `equal_range()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::multiset<_Key, _Compare, _Alloc>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 883 of file `stl_multiset.h`.

**4.733.3.15** `equal_range()` [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset<_Key, _Compare, _Alloc>::equal_range (
 const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 889 of file stl\_multiset.h.

#### 4.733.3.16 equal\_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (
 const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_
tr(__x))) [inline]
```

Finds a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 895 of file stl\_multiset.h.

#### 4.733.3.17 erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 639 of file `stl_multiset.h`.

**4.733.3.18 erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 669 of file `stl_multiset.h`.

**4.733.3.19 erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a `[first,last)` range of elements from a multiset.



## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 691 of file `stl_multiset.h`.

## 4.733.3.20 find() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a set.

## Parameters

|                  |                        |
|------------------|------------------------|
| <code>↵</code>   | Element to be located. |
| <code>__x</code> |                        |

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 775 of file `stl_multiset.h`.

## 4.733.3.21 find() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↵</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 779 of file `stl_multiset.h`.

**4.733.3.22 find()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::find (
 const _Kt & __x) -> decltype(iterator [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↵</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 785 of file `stl_multiset.h`.

**4.733.3.23 find()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::find (
 const _Kt & __x) const -> decltype(const_iterator [inline]
```

Tries to locate an element in a set.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <code>_↵</code> | Element to be located. |
| <code>_X</code> |                        |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 791 of file `stl_multiset.h`.

## 4.733.3.24 get\_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the memory allocation object.

Definition at line 331 of file `stl_multiset.h`.

## 4.733.3.25 insert() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts an element into the multiset.

## Parameters

|                 |                         |
|-----------------|-------------------------|
| <code>_↵</code> | Element to be inserted. |
| <code>_X</code> |                         |

## Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Insertion requires logarithmic time.

Definition at line 502 of file `stl_multiset.h`.

Referenced by `std::multiset< _Key, _Compare, _Alloc >::insert()`.

#### 4.733.3.26 `insert()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Inserts an element into the multiset.

##### Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

##### Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a `std::set` the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 532 of file `stl_multiset.h`.

#### 4.733.3.27 `insert()` [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _InputIterator >
void std::multiset< _Key, _Compare, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that tries to insert a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 551 of file `stl_multiset.h`.

## 4.733.3.28 insert() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the multiset.

## Parameters

|                |                                                                                    |
|----------------|------------------------------------------------------------------------------------|
| <code>↵</code> | A <code>std::initializer_list&lt;value_type&gt;</code> of elements to be inserted. |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>↵</code> |                                                                                    |
| <code>/</code> |                                                                                    |

Complexity similar to that of the range constructor.

Definition at line 563 of file `stl_multiset.h`.

References `std::multiset< _Key, _Compare, _Alloc >::insert()`.

## 4.733.3.29 key\_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object.

Definition at line 323 of file `stl_multiset.h`.

## 4.733.3.30 lower\_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 810 of file `stl_multiset.h`.

**4.733.3.31 lower\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 814 of file `stl_multiset.h`.

**4.733.3.32 lower\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) -> decltype(iterator(_M.t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|       |                    |
|-------|--------------------|
| $\_k$ | Key to be located. |
| $\_x$ |                    |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 820 of file stl\_multiset.h.

## 4.733.3.33 lower\_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) const -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|       |                    |
|-------|--------------------|
| $\_k$ | Key to be located. |
| $\_x$ |                    |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 826 of file stl\_multiset.h.

## 4.733.3.34 max\_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 420 of file stl\_multiset.h.

**4.733.3.35 operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (
 const multiset< _Key, _Compare, _Alloc > &) [default]
```

Multiset assignment operator.

Whether the allocator is copied depends on the allocator traits.

**4.733.3.36 operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (
 multiset< _Key, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

**4.733.3.37 operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
multiset& std::multiset< _Key, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Multiset list assignment operator.

**Parameters**

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills a multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

Definition at line 312 of file stl\_multiset.h.



**4.733.3.38 rbegin()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 358 of file stl\_multiset.h.

**4.733.3.39 rend()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::multiset<_Key, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

Definition at line 367 of file stl\_multiset.h.

**4.733.3.40 size()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
size_type std::multiset<_Key, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the set.

Definition at line 415 of file stl\_multiset.h.

**4.733.3.41 swap()**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::multiset<_Key, _Compare, _Alloc >::swap (
 multiset<_Key, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another multiset.

**Parameters**

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__x</code> | A multiset of the same element and allocator types. |
|------------------|-----------------------------------------------------|

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 437 of file `stl_multiset.h`.

#### 4.733.3.42 `upper_bound()` [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 840 of file `stl_multiset.h`.

#### 4.733.3.43 `upper_bound()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 844 of file stl\_multiset.h.

#### 4.733.3.44 upper\_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

##### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 850 of file stl\_multiset.h.

#### 4.733.3.45 upper\_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

##### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 856 of file stl\_multiset.h.

4.733.3.46 `value_comp()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp () const [inline]
```

Returns the comparison object.

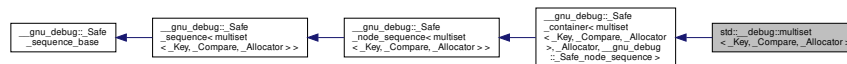
Definition at line 327 of file `stl_multiset.h`.

The documentation for this class was generated from the following file:

- [stl\\_multiset.h](#)

4.734 `std::__debug::multiset< _Key, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::multiset< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_const_iterator, multiset >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::Safe_iterator< _Base_iterator, multiset >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- **multiset** (const [multiset](#) &)=default
- **multiset** ([multiset](#) &&)=default
- **multiset** ([initializer\\_list](#)< value\_type > \_\_l, const \_Compare &\_\_comp=\_Compare(), const allocator\_type &\_\_a=allocator\_type())
- **multiset** (const allocator\_type &\_\_a)
- **multiset** (const [multiset](#) &\_\_m, const allocator\_type &\_\_a)
- **multiset** ([multiset](#) &&\_\_m, const allocator\_type &\_\_a) noexcept(noexcept([\\_Base](#)([std::move](#)(\_\_m.\_M\_base()), [\\_\\_a](#))))
- **multiset** ([initializer\\_list](#)< value\_type > \_\_l, const allocator\_type &\_\_a)
- [template](#)<typename \_InputIterator >  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a)
- **multiset** (const \_Compare &\_\_comp, const \_Allocator &\_\_a=\_Allocator())
- [template](#)<typename \_InputIterator >  
**multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_comp=\_Compare(), const \_Allocator &\_\_a=\_Allocator())
- **multiset** (const [\\_Base](#) &\_\_x)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- void [\\_M\\_transfer\\_from\\_if](#) (\_Safe\_sequence &\_\_from, \_Predicate \_\_pred)
- **iterator begin** () noexcept
- [const\\_iterator begin](#) () const noexcept
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_iterator cend](#) () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator crbegin](#) () const noexcept
- [const\\_reverse\\_iterator crend](#) () const noexcept
- [template](#)<typename... \_Args>  
[iterator emplace](#) (\_Args &&... \_\_args)
- [template](#)<typename... \_Args>  
[iterator emplace\\_hint](#) ([const\\_iterator](#) \_\_pos, \_Args &&... \_\_args)
- **iterator end** () noexcept
- [const\\_iterator end](#) () const noexcept
- [std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const key\_type &\_\_x)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const key\_type &\_\_x) const
- [template](#)<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[std::pair](#)< [iterator](#), [iterator](#) > **equal\_range** (const \_Kt &\_\_x)
- [template](#)<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > **equal\_range** (const \_Kt &\_\_x) const
- \_GLIBCXX\_ABI\_TAG\_CXX11 **iterator erase** ([const\\_iterator](#) \_\_position)
- size\_type **erase** (const key\_type &\_\_x)
- \_GLIBCXX\_ABI\_TAG\_CXX11 **iterator erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
- **iterator find** (const key\_type &\_\_x)
- [const\\_iterator find](#) (const key\_type &\_\_x) const
- [template](#)<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator find](#) (const \_Kt &\_\_x)
- [template](#)<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator find](#) (const \_Kt &\_\_x) const

- [iterator insert](#) (const value\_type &\_\_x)
- [iterator insert](#) (value\_type &&\_\_x)
- [iterator insert](#) (const\_iterator \_\_position, const value\_type &\_\_x)
- [iterator insert](#) (const\_iterator \_\_position, value\_type &&\_\_x)
- template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< value\_type > \_\_l)
- [iterator lower\\_bound](#) (const key\_type &\_\_x)
- [const\\_iterator lower\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator lower\\_bound](#) (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator lower\\_bound](#) (const \_Kt &\_\_x) const
- [multiset & operator=](#) (const [multiset](#) &)=default
- [multiset & operator=](#) ([multiset](#) &&)=default
- [multiset & operator=](#) (initializer\_list< value\_type > \_\_l)
- [reverse\\_iterator rbegin](#) () noexcept
- [const\\_reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () noexcept
- [const\\_reverse\\_iterator rend](#) () const noexcept
- void [swap](#) ([multiset](#) &\_\_x) noexcept(*/\*conditional \*/*)
- [iterator upper\\_bound](#) (const key\_type &\_\_x)
- [const\\_iterator upper\\_bound](#) (const key\_type &\_\_x) const
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[iterator upper\\_bound](#) (const \_Kt &\_\_x)
- template<typename \_Kt, typename \_Req = typename \_\_has\_is\_transparent<\_Compare, \_Kt>::type>  
[const\\_iterator upper\\_bound](#) (const \_Kt &\_\_x) const

## Public Attributes

- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_const\\_iterators](#)
- [\\_Safe\\_iterator\\_base](#) \* [\\_M\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex & \\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container & \\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x) noexcept

## Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT >  
class [\\_\\_gnu\\_debug::Safe\\_iterator](#)

#### 4.734.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::multiset<_Key, _Compare, _Allocator >
```

Class std::multiset with safety/checking/debug instrumentation.

Definition at line 44 of file multiset.h.

#### 4.734.2 Member Function Documentation

##### 4.734.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

##### 4.734.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

##### 4.734.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map<\_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.734.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.734.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 4.734.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.734.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.734.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< multiset< _Key, _Compare, _Allocator > >::_M_transfer_from_if
(
 _Safe_sequence< multiset< _Key, _Compare, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.



#### 4.734.3 Member Data Documentation

##### 4.734.3.1 `_M_const_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.734.3.2 `_M_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

##### 4.734.3.3 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

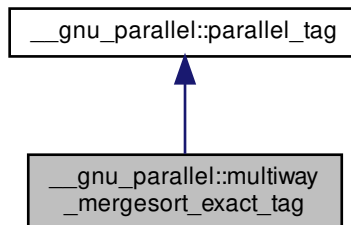
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [multiset.h](#)

#### 4.735 \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag:



##### Public Member Functions

- **multiway\_mergesort\_exact\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

##### 4.735.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

Definition at line 137 of file tags.h.

##### 4.735.2 Member Function Documentation

###### 4.735.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

###### 4.735.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

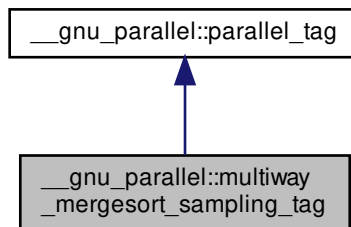
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.736 \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag:



## Public Member Functions

- **multiway\_mergesort\_sampling\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 4.736.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

Definition at line 146 of file tags.h.

## 4.736.2 Member Function Documentation

#### 4.736.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

#### 4.736.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

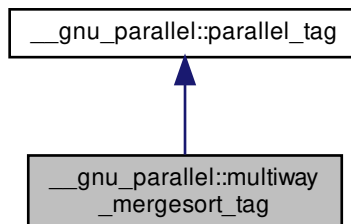
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.737 \_\_gnu\_parallel::multiway\_mergesort\_tag Struct Reference

Inheritance diagram for `__gnu_parallel::multiway_mergesort_tag`:



## Public Member Functions

- **multiway\_mergesort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

## 4.737.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

Definition at line 128 of file tags.h.

## 4.737.2 Member Function Documentation

## 4.737.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

## 4.737.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.738 `std::mutex` Class Reference

Inherits `std::__mutex_base`.

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `mutex (const mutex &)=delete`
- `void lock ()`
- `native_handle_type native_handle () noexcept`
- `mutex & operator= (const mutex &)=delete`
- `bool try_lock () noexcept`
- `void unlock ()`

### Private Types

- `typedef __gthread_mutex_t __native_type`

### Private Attributes

- `__native_type _M_mutex`

#### 4.738.1 Detailed Description

The standard mutex type.

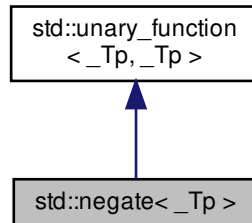
Definition at line 83 of file `std_mutex.h`.

The documentation for this class was generated from the following file:

- [std\\_mutex.h](#)

## 4.739 std::negate< \_Tp > Struct Template Reference

Inheritance diagram for std::negate< \_Tp >:



### Public Types

- typedef \_Tp [argument\\_type](#)
- typedef \_Tp [result\\_type](#)

### Public Member Functions

- constexpr \_Tp **operator()** (const \_Tp &\_\_x) const

#### 4.739.1 Detailed Description

```
template<typename _Tp>
struct std::negate< _Tp >
```

One of the [math functors](#).

Definition at line 162 of file stl\_function.h.

#### 4.739.2 Member Typedef Documentation

##### 4.739.2.1 argument\_type

```
typedef _Tp std::unary_function< _Tp , _Tp >::argument_type [inherited]
```

[argument\\_type](#) is the type of the argument

Definition at line 108 of file stl\_function.h.

#### 4.739.2.2 `result_type`

```
typedef _Tp std::unary_function< _Tp , _Tp >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.740 `std::negate< void >` Struct Template Reference

#### Public Types

- typedef `__is_transparent` **`is_transparent`**

#### Public Member Functions

- `template<typename _Tp >`  
`constexpr auto operator()` (`_Tp &&__t`) `const noexcept(noexcept(-std::forward< _Tp >(__t))) -> decltype(-std::forward< _Tp >(__t))`

#### 4.740.1 Detailed Description

```
template<>
struct std::negate< void >
```

One of the [math functors](#).

Definition at line 305 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.741 `std::negative_binomial_distribution< _IntType >` Class Template Reference

#### Classes

- struct [param\\_type](#)



## Public Types

- typedef \_IntType [result\\_type](#)

## Public Member Functions

- **negative\_binomial\_distribution** (\_IntType \_\_k, double \_\_p=0.5)
- **negative\_binomial\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_IntType **k** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- double **p** () const
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & **operator<<** ([std::basic\\_ostream](#)< \_CharT, \_Traits > &\_\_os, const [std::negative\\_binomial\\_distribution](#)< \_IntType1 > &\_\_x)
- bool **operator==** (const [negative\\_binomial\\_distribution](#) &\_\_d1, const [negative\\_binomial\\_distribution](#) &\_\_d2)
- template<typename \_IntType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & **operator>>** ([std::basic\\_istream](#)< \_CharT, \_Traits > &\_\_is, [std::negative\\_binomial\\_distribution](#)< \_IntType1 > &\_\_x)

## 4.741.1 Detailed Description

```
template<typename _IntType = int>
class std::negative_binomial_distribution< _IntType >
```

A [negative\\_binomial\\_distribution](#) random number distribution.

The formula for the negative binomial probability mass function is  $p(i) = \binom{n}{i} p^i (1 - p)^{t-i}$  where  $t$  and  $p$  are the parameters of the distribution.

Definition at line 4188 of file random.h.

#### 4.741.2 Member Typedef Documentation

##### 4.741.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::negative_binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 4191 of file random.h.

#### 4.741.3 Member Function Documentation

##### 4.741.3.1 k()

```
template<typename _IntType = int>
_IntType std::negative_binomial_distribution< _IntType >::k () const [inline]
```

Return the  $k$  parameter of the distribution.

Definition at line 4255 of file random.h.

##### 4.741.3.2 max()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4291 of file random.h.

References `std::numeric_limits< _Tp >::max()`.

##### 4.741.3.3 min()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4284 of file random.h.

## 4.741.3.4 operator()

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng)
```

Generating functions.

Definition at line 1112 of file bits/random.tcc.

## 4.741.3.5 p()

```
template<typename _IntType = int>
double std::negative_binomial_distribution< _IntType >::p () const [inline]
```

Return the  $p$  parameter of the distribution.

Definition at line 4262 of file random.h.

## 4.741.3.6 param() [1/2]

```
template<typename _IntType = int>
param_type std::negative_binomial_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4269 of file random.h.

## 4.741.3.7 param() [2/2]

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4277 of file random.h.

#### 4.741.3.8 reset()

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Definition at line 4248 of file random.h.

References `std::gamma_distribution< _RealType >::reset()`.

### 4.741.4 Friends And Related Function Documentation

#### 4.741.4.1 operator<<

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::negative_binomial_distribution< _IntType1 > & __x) [friend]
```

Inserts a `negative_binomial_distribution` random number distribution `__x` into the output stream `__os`.

##### Parameters

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                         |
| <code>__x</code>  | A <code>negative_binomial_distribution</code> random number distribution. |

##### Returns

The output stream with the state of `__x` inserted or in an error state.

#### 4.741.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
 const negative_binomial_distribution< _IntType > & __d1,
 const negative_binomial_distribution< _IntType > & __d2) [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 4340 of file random.h.

## 4.741.4.3 operator&gt;&gt;

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::negative_binomial_distribution< _IntType1 > & __x) [friend]
```

Extracts a `negative_binomial_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                              |
| <code>__x</code>  | A <code>negative_binomial_distribution</code> random number generator engine. |

## Returns

The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.742 std::nested\_exception Class Reference

## Public Member Functions

- **nested\_exception** (const [nested\\_exception](#) &) noexcept=default
- **exception\_ptr nested\_ptr** () const noexcept
- **nested\_exception & operator=** (const [nested\\_exception](#) &) noexcept=default
- void **rethrow\_nested** () const

## 4.742.1 Detailed Description

Exception class with `exception_ptr` data member.

Definition at line 52 of file `nested_exception.h`.

The documentation for this class was generated from the following file:

- [nested\\_exception.h](#)

#### 4.743 `__gnu_cxx::limit_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::limit_condition::adjustor_base`.

##### 4.743.1 Detailed Description

Never enter the condition.

Definition at line 446 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.744 `__gnu_cxx::random_condition::never_adjustor` Struct Reference

Inherits `__gnu_cxx::random_condition::adjustor_base`.

##### 4.744.1 Detailed Description

Never enter the condition.

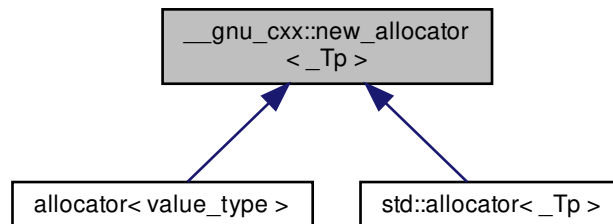
Definition at line 527 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.745 `__gnu_cxx::new_allocator<_Tp>` Class Template Reference

Inheritance diagram for `__gnu_cxx::new_allocator<_Tp>`:



## Public Types

- `typedef const _Tp * const_pointer`
- `typedef const _Tp & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Tp * pointer`
- `typedef std::true\_type propagate_on_container_move_assignment`
- `typedef _Tp & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `constexpr new_allocator (const new\_allocator &) noexcept`
- `template<typename _Tp1 >  
constexpr new_allocator (const new\_allocator<_Tp1 > &) noexcept`
- `pointer address (reference __x) const noexcept`
- `const_pointer address (const_reference __x) const noexcept`
- `_Tp * allocate (size_type __n, const void **=static_cast< const void * >(0))`
- `template<typename _Up, typename... _Args>  
void construct (_Up * __p, _Args &&... __args) noexcept(std::is\_nothrow\_constructible<_Up, _Args... >::value)`
- `void deallocate (_Tp * __p, size_type __t)`
- `template<typename _Up >  
void destroy (_Up * __p) noexcept(std::is\_nothrow\_destructible<_Up >::value)`
- `size_type max_size () const noexcept`

## Friends

- `template<typename _Up >  
constexpr bool operator!= (const new\_allocator &, const new\_allocator<_Up > &) noexcept`
- `template<typename _Up >  
constexpr bool operator== (const new\_allocator &, const new\_allocator<_Up > &) noexcept`

## 4.745.1 Detailed Description

```
template<typename _Tp>
class __gnu_cxx::new_allocator<_Tp>
```

An allocator that uses global new, as per [20.4].

This is precisely the allocator defined in the C++ Standard.

- all allocation calls operator new
- all deallocation calls operator delete

## Template Parameters

|                   |                           |
|-------------------|---------------------------|
| <code>__Tp</code> | Type of allocated object. |
|-------------------|---------------------------|

Definition at line 55 of file `new_allocator.h`.

The documentation for this class was generated from the following file:

- [new\\_allocator.h](#)

#### 4.746 `__gnu_pbds::detail::no_throw_copies< Key, Mapped >` Struct Template Reference

## Public Types

- `typedef integral_constant< int, __simple > indicator`

## Static Public Attributes

- `static const bool __simple`

##### 4.746.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
```

Primary template.

Definition at line 61 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.747 `__gnu_pbds::detail::no_throw_copies< Key, null_type >` Struct Template Reference

## Public Types

- `typedef integral_constant< int, is_simple< Key >::value > indicator`



## 4.747.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
```

Specialization.

Definition at line 70 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.748 `std::normal_distribution<_RealType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- [normal\\_distribution](#) ([result\\_type](#) \_\_mean, [result\\_type](#) \_\_stddev=[result\\_type](#)(1))
- **[normal\\_distribution](#)** (const [param\\_type](#) &\_\_p)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_ForwardIterator`, typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** (`_ForwardIterator` \_\_f, `_ForwardIterator` \_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename `_UniformRandomNumberGenerator` >  
void **[generate](#)** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, `_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) [max](#) () const
- `_RealType` [mean](#) () const
- [result\\_type](#) [min](#) () const
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &\_\_urng)
- template<typename `_UniformRandomNumberGenerator` >  
[result\\_type](#) [operator\(\)](#) (`_UniformRandomNumberGenerator` &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) [param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()
- `_RealType` [stddev](#) () const

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::normal_distribution< _RealType1 > &__x)`
- `template<typename _RealType1 >`  
`bool operator==(const std::normal_distribution< _RealType1 > &__d1, const std::normal_distribution< _RealType1 > &__d2)`
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`std::normal_distribution< _RealType1 > &__x)`

### 4.748.1 Detailed Description

```
template<typename _RealType = double>
class std::normal_distribution< _RealType >
```

A normal continuous distribution for random numbers.

The formula for the normal probability density function is

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma^2}}$$

Definition at line 1970 of file random.h.

### 4.748.2 Member Typedef Documentation

#### 4.748.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 1973 of file random.h.

### 4.748.3 Constructor & Destructor Documentation

#### 4.748.3.1 normal\_distribution()

```
template<typename _RealType = double>
std::normal_distribution< _RealType >::normal_distribution (
 result_type __mean,
 result_type __stddev = result_type(1)) [inline], [explicit]
```

Constructs a normal distribution with parameters *mean* and standard deviation.

Definition at line 2023 of file random.h.

#### 4.748.4 Member Function Documentation

##### 4.748.4.1 max()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 2080 of file random.h.

##### 4.748.4.2 mean()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::mean () const [inline]
```

Returns the mean of the distribution.

Definition at line 2044 of file random.h.

##### 4.748.4.3 min()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 2073 of file random.h.

**4.748.4.4 operator()** [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::normal_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 2088 of file random.h.

Referenced by std::normal\_distribution< result\_type >::operator()().

**4.748.4.5 operator()** [2/2]

```
template<typename _RealType >
template<typename _UniformRandomNumberGenerator >
normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator()
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

Polar method due to Marsaglia.

Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.

Definition at line 1770 of file bits/random.tcc.

**4.748.4.6 param()** [1/2]

```
template<typename _RealType = double>
param_type std::normal_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 2058 of file random.h.

**4.748.4.7 param()** [2/2]

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 2066 of file random.h.

## 4.748.4.8 reset()

```
template<typename _RealType = double>
void std::normal_distribution<_RealType>::reset () [inline]
```

Resets the distribution state.

Definition at line 2037 of file random.h.

Referenced by `std::lognormal_distribution<_RealType>::reset()`, `std::gamma_distribution<result_type>::reset()`, `std::student_t_distribution<_RealType>::reset()`, `std::binomial_distribution<_IntType>::reset()`, and `std::poisson_distribution<_IntType>::reset()`.

## 4.748.4.9 stddev()

```
template<typename _RealType = double>
_RealType std::normal_distribution<_RealType>::stddev () const [inline]
```

Returns the standard deviation of the distribution.

Definition at line 2051 of file random.h.

## 4.748.5 Friends And Related Function Documentation

## 4.748.5.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream<_CharT, _Traits> & __os,
 const std::normal_distribution<_RealType1> & __x) [friend]
```

Inserts a `normal_distribution` random number distribution `__x` into the output stream `__os`.

**Parameters**

|                   |                                                   |
|-------------------|---------------------------------------------------|
| <code>__os</code> | An output stream.                                 |
| <code>__x</code>  | A normal_distribution random number distribution. |

**Returns**

The output stream with the state of `__x` inserted or in an error state.

**4.748.5.2 operator==**

```
template<typename _RealType = double>
template<typename _RealType1 >
bool operator== (
 const std::normal_distribution< _RealType1 > & __d1,
 const std::normal_distribution< _RealType1 > & __d2) [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

**4.748.5.3 operator>>**

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::normal_distribution< _RealType1 > & __x) [friend]
```

Extracts a normal\_distribution random number distribution `__x` from the input stream `__is`.

**Parameters**

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <code>__is</code> | An input stream.                                      |
| <code>__x</code>  | A normal_distribution random number generator engine. |

**Returns**

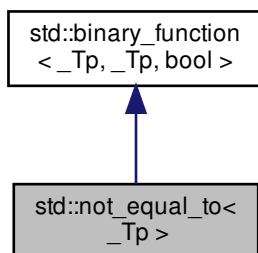
The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.749 std::not\_equal\_to< \_Tp > Struct Template Reference

Inheritance diagram for std::not\_equal\_to< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `bool` **operator()** (`const _Tp &__x`, `const _Tp &__y`) `const`

#### 4.749.1 Detailed Description

```
template<typename _Tp>
struct std::not_equal_to< _Tp >
```

One of the [comparison functors](#).

Definition at line 334 of file `stl_function.h`.

#### 4.749.2 Member Typedef Documentation

#### 4.749.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.749.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.749.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.750 std::not\_equal\_to< void > Struct Template Reference

#### Public Types

- typedef \_\_is\_transparent **is\_transparent**

#### Public Member Functions

- template<typename \_Tp, typename \_Up >  
constexpr auto **operator()** (\_Tp &&\_\_t, \_Up &&\_\_u) const noexcept(noexcept(std::forward< \_Tp >(\_\_t) !=std::forward< \_Up >(\_\_u))) -> decltype(std::forward< \_Tp >(\_\_t) !=std::forward< \_Up >(\_\_u))



## 4.750.1 Detailed Description

```
template<>
struct std::not_equal_to< void >
```

One of the [comparison functors](#).

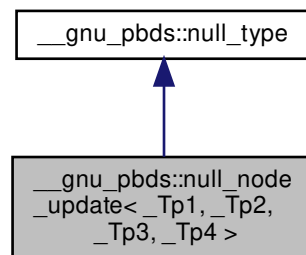
Definition at line 502 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.751 \_\_gnu\_pbds::null\_node\_update&lt; \_Tp1, \_Tp2, \_Tp3, \_Tp4 &gt; Struct Template Reference

Inheritance diagram for `__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >`:



## 4.751.1 Detailed Description

```
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
```

A null node updatator, indicating that no node updates are required.

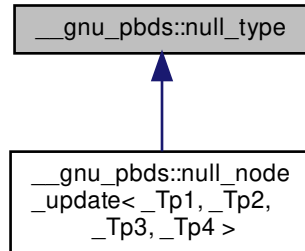
Definition at line 214 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.752 `__gnu_pbds::null_type` Struct Reference

Inheritance diagram for `__gnu_pbds::null_type`:



##### 4.752.1 Detailed Description

Represents no type, or absence of type, for template tricks.

In a mapped-policy, indicates that an associative container is a set.

In a list-update policy, indicates that each link does not need metadata.

In a hash policy, indicates that the combining hash function is actually a ranged hash function.

In a probe policy, indicates that the combining probe function is actually a ranged probe function.

Definition at line 210 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.753 `std::experimental::fundamentals_v1::nullopt_t` Struct Reference

##### Public Types

- `enum _Construct { _Token }`

##### Public Member Functions

- `constexpr nullopt_t (_Construct)`

## 4.753.1 Detailed Description

Tag type to disengage optional objects.

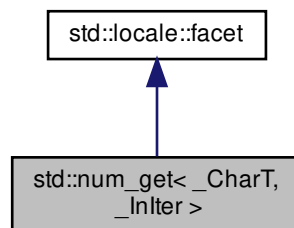
Definition at line 81 of file experimental/optional.

The documentation for this struct was generated from the following file:

- [experimental/optional](#)

## 4.754 std::num\_get&lt; \_CharT, \_InIter &gt; Class Template Reference

Inheritance diagram for std::num\_get< \_CharT, \_InIter >:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef \_InIter [iter\\_type](#)

## Public Member Functions

- [num\\_get](#) (size\_t \_\_refs=0)
- template<typename \_ValueT >  
\_GLIBCXX\_DEFAULT\_ABI\_TAG \_InIter **M\_extract\_int** (\_InIter \_\_beg, \_InIter \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, \_ValueT &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, bool &\_\_v) const
- [iter\\_type get](#) ([iter\\_type](#) \_\_in, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, void \*&\_\_v) const

- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, float & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, double & __v) const`
- `iter_type get (iter_type __in, iter_type __end, ios_base & __io, ios_base::iostate & __err, long double & __v) const`

#### Static Public Attributes

- static `locale::id id`

#### Protected Member Functions

- virtual `~num_get ()`
- `_GLIBCXX_DEFAULT_ABI_TAG iter_type _M_extract_float (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const`
- `template<typename _ValueT > _GLIBCXX_DEFAULT_ABI_TAG iter_type _M_extract_int (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const`
- `template<typename _CharT2 > _gnu_cxx::__enable_if< _is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 *, size_t __len, _CharT2 __c) const`
- `template<typename _CharT2 > _gnu_cxx::__enable_if< !_is_char< _CharT2 >::__value, int >::__type _M_find (const _CharT2 * __zero, size_t __len, _CharT2 __c) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned short & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned int & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, long long & __v) const`
- virtual `iter_type do_get (iter_type __beg, iter_type __end, ios_base & __io, ios_base::iostate & __err, unsigned long long & __v) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const`
- virtual `iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const`

## Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.754.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >
```

Primary class template num\_get.

This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.

The num\_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num\_get facet.

Definition at line 1952 of file locale\_facets.h.

## 4.754.2 Member Typedef Documentation

## 4.754.2.1 char\_type

```
template<typename _CharT , typename _InIter >
typedef _CharT std::num_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 1958 of file locale\_facets.h.

## 4.754.2.2 iter\_type

```
template<typename _CharT , typename _InIter >
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 1959 of file locale\_facets.h.

#### 4.754.3 Constructor & Destructor Documentation

##### 4.754.3.1 num\_get()

```
template<typename _CharT , typename _InIter >
std::num_get< _CharT, _InIter >::num_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

##### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 1973 of file locale\_facets.h.

##### 4.754.3.2 ~num\_get()

```
template<typename _CharT , typename _InIter >
virtual std::num_get< _CharT, _InIter >::~~num_get () [inline], [protected], [virtual]
```

Destructor.

Definition at line 2145 of file locale\_facets.h.

#### 4.754.4 Member Function Documentation

##### 4.754.4.1 do\_get() [1/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 bool & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 595 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::boolalpha`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::flags()`, and `std::ios_base::goodbit`.

Referenced by `std::num_get< _CharT, _InIter >::get()`.

## 4.754.4.2 do\_get() [2/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

## See also

`get()` for more details.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2215 of file locale\_facets.h.

**4.754.4.3 do\_get()** [3/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned short & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2220 of file locale\_facets.h.

**4.754.4.4 do\_get()** [4/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
```



```

ios_base & __io,
ios_base::iostate & __err,
unsigned int & __v) const [inline], [protected], [virtual]

```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

Returns

Iterator after reading.

Definition at line 2225 of file locale\_facets.h.

#### 4.754.4.5 do\_get() [5/11]

```

template<typename _CharT , typename _InIter >
virtual iter_type std::num_get<_CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long & __v) const [inline], [protected], [virtual]

```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2230 of file `locale_facets.h`.

**4.754.4.6 do\_get()** [6/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2236 of file `locale_facets.h`.

## 4.754.4.7 do\_get() [7/11]

```
template<typename _CharT , typename _InIter >
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long long & __v) const [inline], [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2241 of file locale\_facets.h.

## 4.754.4.8 do\_get() [8/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 float & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable v. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 691 of file `locale_facets.tcc`.

References `std::basic_string<_CharT, _Traits, _Alloc>::c_str()`, `std::ios_base::eofbit`, and `std::basic_string<_CharT, _Traits, _Alloc>::reserve()`.

**4.754.4.9 do\_get()** [9/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get<_CharT, _InIter>::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 double & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable `v`. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 706 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

**4.754.4.10 do\_get()** [10/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

get() for more details.

**Parameters**

|              |                             |
|--------------|-----------------------------|
| <i>__beg</i> | Start of input stream.      |
| <i>__end</i> | End of input stream.        |
| <i>__io</i>  | Source of locale and flags. |
| <i>__err</i> | Error flags to set.         |
| <i>__v</i>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 738 of file locale\_facets.tcc.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::c\_str(), std::ios\_base::eofbit, and std::basic\_string< \_CharT, \_Traits, \_Alloc >::reserve().

**4.754.4.11 do\_get()** [11/11]

```
template<typename _CharT , typename _InIter >
_InIter std::num_get< _CharT, _InIter >::do_get (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 void *& __v) const [protected], [virtual]
```

Numeric parsing.

Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

**See also**

`get()` for more details.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__beg</code> | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 753 of file `locale_facets.tcc`.

References `std::ios_base::basefield`, `std::ios_base::flags()`, and `std::ios_base::hex`.

**4.754.4.12 get()** [1/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 bool & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the bool *v*. It does so by calling `num_get::do_get()`.

If `ios_base::boolalpha` is set, attempts to read `ctype<CharT>::truenamename()` or `ctype<CharT>::falsename()`. Sets `v` to true or false if successful. Sets `err` to `ios_base::failbit` if reading the string fails. Sets `err` to `ios_base::eofbit` if the stream is emptied.

If `ios_base::boolalpha` is not set, proceeds as with reading a long, except if the value is 1, sets `v` to true, if the value is 0, sets `v` to false, and otherwise set `err` to `ios_base::failbit`.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 1999 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

**4.754.4.13 get()** [2/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in `io`.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `numpunct::grouping()` and `numpunct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |



**Returns**

Iterator after reading.

Definition at line 2036 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

**4.754.4.14 get()** [3/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned short & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2041 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

**4.754.4.15** `get()` [4/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned int & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2046 of file `locale_facets.h`.

References `std::num_get< _CharT, _InIter >::do_get()`.

**4.754.4.16** `get()` [5/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
```

```

ios_base & __io,
ios_base::iostate & __err,
unsigned long & __v) const [inline]

```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of io.flags() & ios\_base::basefield. If equal to ios\_base::oct, parses like the scanf o specifier. Else if equal to ios\_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2051 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

#### 4.754.4.17 get() [6/11]

```

template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long long & __v) const [inline]

```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2057 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

#### 4.754.4.18 `get()` [7/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get<_CharT, _InIter>::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 unsigned long long & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling `num_get::do_get()`.

Parsing is affected by the flag settings in *io*.

The basic parse is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, parses like the `scanf` `o` specifier. Else if equal to `ios_base::hex`, parses like `X` specifier. Else if `basefield` equal to 0, parses like the `i` specifier. Otherwise, parses like `d` for signed and `u` for unsigned types. The matching type length modifier is also used.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2062 of file locale\_facets.h.

References `std::num_get< _CharT, _InIter >::do_get()`.

## 4.754.4.19 get() [8/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 float & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2096 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

**4.754.4.20 get()** [9/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable `v`. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf g` specifier. The matching type length modifier is also used.

The decimal point character used is `num_punct::decimal_point()`. Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

If parsing the string yields a valid value for `v`, `v` is set. Otherwise, sets `err` to `ios_base::failbit` and leaves `v` unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

**Parameters**

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

**Returns**

Iterator after reading.

Definition at line 2101 of file locale\_facets.h.

References `std::num_get<_CharT, _InIter >::do_get()`.

## 4.754.4.21 get() [10/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 long double & __v) const [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num\_get::do\_get().

The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.

The decimal point character used is numpunct::decimal\_point(). Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands\_sep(). If the pattern of digit groups isn't consistent, sets err to ios\_base::failbit.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios\_base::failbit and leaves *v* unaltered. Sets err to ios\_base::eofbit if the stream is emptied.

## Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

## Returns

Iterator after reading.

Definition at line 2106 of file locale\_facets.h.

References std::num\_get< \_CharT, \_InIter >::do\_get().

## 4.754.4.22 get() [11/11]

```
template<typename _CharT , typename _InIter >
iter_type std::num_get< _CharT, _InIter >::get (
 iter_type __in,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 void *& __v) const [inline]
```

Numeric parsing.

Parses the input stream into the pointer variable *v*. It does so by calling `num_get::do_get()`.

The input characters are parsed like the `scanf p` specifier.

Digit grouping is interpreted according to `num_punct::grouping()` and `num_punct::thousands_sep()`. If the pattern of digit groups isn't consistent, sets `err` to `ios_base::failbit`.

Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets `err` to `ios_base::failbit` and leaves *v* unaltered. Sets `err` to `ios_base::eofbit` if the stream is emptied.

#### Parameters

|                    |                             |
|--------------------|-----------------------------|
| <code>__in</code>  | Start of input stream.      |
| <code>__end</code> | End of input stream.        |
| <code>__io</code>  | Source of locale and flags. |
| <code>__err</code> | Error flags to set.         |
| <code>__v</code>   | Value to format and insert. |

#### Returns

Iterator after reading.

Definition at line 2139 of file `locale_facets.h`.

References `std::num_get<_CharT, _InIter>::do_get()`.

### 4.754.5 Member Data Documentation

#### 4.754.5.1 `id`

```
template<typename _CharT , typename _InIter >
locale::id std::num_get< _CharT, _InIter >::id [static]
```

Numpunct facet `id`.

Definition at line 1963 of file `locale_facets.h`.

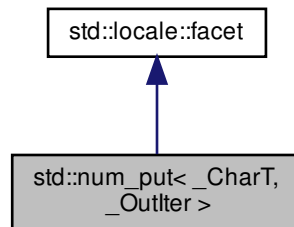
The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)



## 4.755 std::num\_put&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for std::num\_put< \_CharT, \_Outlter >:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef \_Outlter [iter\\_type](#)

## Public Member Functions

- [num\\_put](#) (size\_t \_\_refs=0)
- template<typename \_ValueT >  
\_Outlter [\\_M\\_insert\\_float](#) (\_Outlter \_\_s, [ios\\_base](#) &\_\_io, \_CharT \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- template<typename \_ValueT >  
\_Outlter [\\_M\\_insert\\_int](#) (\_Outlter \_\_s, [ios\\_base](#) &\_\_io, \_CharT \_\_fill, \_ValueT \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, bool \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const void \*\_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, long \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, unsigned long \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, long long \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, unsigned long long \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, double \_\_v) const
- [iter\\_type](#) put ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, long double \_\_v) const

## Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual `~num_put()`
- void `_M_group_float` (const char \* \_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, const char\_type \* \_\_p, char\_type \* \_\_new, char\_type \* \_\_cs, int & \_\_len) const
- void `_M_group_int` (const char \* \_\_grouping, size\_t \_\_grouping\_size, char\_type \_\_sep, ios\_base & \_\_io, char\_type \* \_\_new, char\_type \* \_\_cs, int & \_\_len) const
- template<typename \_ValueT >  
iter\_type `_M_insert_float` (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, char \_\_mod, \_ValueT \_\_v) const
- template<typename \_ValueT >  
iter\_type `_M_insert_int` (iter\_type, ios\_base & \_\_io, char\_type \_\_fill, \_ValueT \_\_v) const
- void `_M_pad` (char\_type \_\_fill, streamsize \_\_w, ios\_base & \_\_io, char\_type \* \_\_new, const char\_type \* \_\_cs, int & \_\_len) const
- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, bool \_\_v) const
- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long \_\_v) const
- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long \_\_v) const
- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, long long \_\_v) const
- virtual iter\_type `do_put` (iter\_type \_\_s, ios\_base & \_\_io, char\_type \_\_fill, unsigned long long \_\_v) const
- virtual iter\_type `do_put` (iter\_type, ios\_base &, char\_type, double) const
- virtual iter\_type `do_put` (iter\_type, ios\_base &, char\_type, long double) const
- virtual iter\_type `do_put` (iter\_type, ios\_base &, char\_type, const void \*) const

## Static Protected Member Functions

- static \_\_c\_locale `_S_clone_c_locale` (\_\_c\_locale & \_\_cloc) throw ()
- static void `_S_create_c_locale` (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale `_S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static \_\_c\_locale `_S_lc_ctype_c_locale` (\_\_c\_locale \_\_cloc, const char \* \_\_s)

### 4.755.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::num_put< _CharT, _Outiter >
```

Primary class template `num_put`.

This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators.

The `num_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `num_put` facet.

Definition at line 2293 of file `locale_facets.h`.

#### 4.755.2 Member Typedef Documentation

##### 4.755.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::num_put< _CharT, _OutIter >::char_type
```

Public typedefs.

Definition at line 2299 of file locale\_facets.h.

##### 4.755.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

Definition at line 2300 of file locale\_facets.h.

#### 4.755.3 Constructor & Destructor Documentation

##### 4.755.3.1 num\_put()

```
template<typename _CharT , typename _OutIter >
std::num_put< _CharT, _OutIter >::num_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 2314 of file locale\_facets.h.

#### 4.755.3.2 ~num\_put()

```
template<typename _CharT, typename _OutIter >
virtual std::num_put< _CharT, _OutIter >::~~num_put () [inline], [protected], [virtual]
```

Destructor.

Definition at line 2493 of file locale\_facets.h.

#### 4.755.4 Member Function Documentation

##### 4.755.4.1 do\_put() [1/8]

```
template<typename _CharT, typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 bool __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

##### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

##### Returns

Iterator after writing.

Definition at line 1106 of file locale\_facets.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::adjustfield`, `std::ios_base::boolalpha`, `std::ios_base::flags()`, `std::ios_base::left`, and `std::ios_base::width()`.

Referenced by `std::num_put< _CharT, _OutIter >::put()`.

## 4.755.4.2 do\_put() [2/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2513 of file locale\_facets.h.

## 4.755.4.3 do\_put() [3/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | Stream to write to. |
|------------------|---------------------|

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2517 of file locale\_facets.h.

**4.755.4.4 do\_put()** [4/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2523 of file locale\_facets.h.

## 4.755.4.5 do\_put() [5/8]

```
template<typename _CharT , typename _OutIter >
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long long __v) const [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2528 of file locale\_facets.h.

## 4.755.4.6 do\_put() [6/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | Stream to write to. |
|------------------|---------------------|

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 1158 of file locale\_facets.tcc.

**4.755.4.7 do\_put()** [7/8]

```
template<typename _CharT, typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long double __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 1172 of file locale\_facets.tcc.



## 4.755.4.8 do\_put() [8/8]

```
template<typename _CharT , typename _OutIter >
_OutIter std::num_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const void * __v) const [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 1179 of file locale\_facets.tcc.

References `std::ios_base::basefield`, `std::ios_base::flags()`, `std::ios_base::hex`, `std::ios_base::showbase`, and `std::ios_base::uppercase`.

## 4.755.4.9 put() [1/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 bool __v) const [inline]
```

Numeric formatting.

Formats the boolean `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

If `ios_base::boolalpha` is set, writes `ctype<CharT>::truename()` or `ctype<CharT>::falsename()`. Otherwise formats `v` as an int.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2332 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

4.755.4.10 `put()` [2/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long __v) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2374 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 4.755.4.11 put() [3/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long __v) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2378 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

4.755.4.12 `put()` [4/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long long __v) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2384 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 4.755.4.13 put() [5/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 unsigned long long __v) const [inline]
```

Numeric formatting.

Formats the integral value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::basefield`. If equal to `ios_base::oct`, formats like the `printf o` specifier. Else if equal to `ios_base::hex`, formats like `x` or `X` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `d`, `ld`, `lld` for signed and `u`, `lu`, `llu` for unsigned values. Note that if both `oct` and `hex` are set, neither will take effect.

If `ios_base::showpos` is set, `'+'` is output before positive values. If `ios_base::showbase` is set, `'0'` precedes octal values (except 0) and `'0[xX]'` precedes hex values.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a `'+'` or `'-'` or after `'0x'` or `'0X'`. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2388 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter>::do_put()`.

4.755.4.14 `put()` [6/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 double __v) const [inline]
```

Numeric formatting.

Formats the floating point value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in `io`.

The basic format is affected by the value of `io.flags()` & `ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf `f` specifier. Else if equal to `ios_base::scientific`, formats like `e` or `E` with `ios_base::uppercase` unset or set respectively. Otherwise, formats like `g` or `G` depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like `g` or `G`.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `num_punct::decimal_point()`. Thousands separators are inserted according to `num_punct::grouping()` and `num_punct::thousands_sep()`.

If `io.width()` is non-zero, enough `fill` characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

## Returns

Iterator after writing.

Definition at line 2437 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

## 4.755.4.15 put() [7/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 long double __v) const [inline]
```

Numeric formatting.

Formats the floating point value *v* and inserts it into a stream. It does so by calling `num_put::do_put()`.

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of `io.flags() & ios_base::floatfield`. If equal to `ios_base::fixed`, formats like the printf *f* specifier. Else if equal to `ios_base::scientific`, formats like *e* or *E* with `ios_base::uppercase` unset or set respectively. Otherwise, formats like *g* or *G* depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like *g* or *G*.

The output precision is given by `io.precision()`. This precision is capped at `numeric_limits::digits10 + 2` (different for double and long double). The default precision is 6.

If `ios_base::showpos` is set, '+' is output before positive values. If `ios_base::showpoint` is set, a decimal point will always be output.

The decimal point character used is `numput::decimal_point()`. Thousands separators are inserted according to `numput::grouping()` and `numput::thousands_sep()`.

If `io.width()` is non-zero, enough *fill* characters are inserted to make the result at least that wide. If `(io.flags() & ios_base::adjustfield) == ios_base::left`, result is padded at the end. If `ios_base::internal`, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |

**Returns**

Iterator after writing.

Definition at line 2441 of file locale\_facets.h.

References `std::num_put<_CharT, _OutIter >::do_put()`.

**4.755.4.16 put()** [8/8]

```
template<typename _CharT , typename _OutIter >
iter_type std::num_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const void * __v) const [inline]
```

Numeric formatting.

Formats the pointer value `v` and inserts it into a stream. It does so by calling `num_put::do_put()`.

This function formats `v` as an unsigned long with `ios_base::hex` and `ios_base::showbase` set.

**Parameters**

|                     |                               |
|---------------------|-------------------------------|
| <code>__s</code>    | Stream to write to.           |
| <code>__io</code>   | Source of locale and flags.   |
| <code>__fill</code> | Char_type to use for filling. |
| <code>__v</code>    | Value to format and insert.   |



**Returns**

Iterator after writing.

Definition at line 2462 of file locale\_facets.h.

References `std::num_put< _CharT, _OutIter >::do_put()`.

**4.755.5 Member Data Documentation****4.755.5.1 id**

```
template<typename _CharT , typename _OutIter >
locale::id std::num_put< _CharT, _OutIter >::id [static]
```

Numpunct facet id.

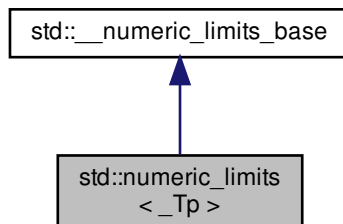
Definition at line 2304 of file locale\_facets.h.

The documentation for this class was generated from the following files:

- [locale\\_facets.h](#)
- [locale\\_facets.tcc](#)

**4.756 std::numeric\_limits< \_Tp > Struct Template Reference**

Inheritance diagram for `std::numeric_limits< _Tp >`:



### Static Public Member Functions

- static constexpr `_Tp` `denorm_min` () noexcept
- static constexpr `_Tp` `epsilon` () noexcept
- static constexpr `_Tp` `infinity` () noexcept
- static constexpr `_Tp` `lowest` () noexcept
- static constexpr `_Tp` `max` () noexcept
- static constexpr `_Tp` `min` () noexcept
- static constexpr `_Tp` `quiet_NaN` () noexcept
- static constexpr `_Tp` `round_error` () noexcept
- static constexpr `_Tp` `signaling_NaN` () noexcept

### Static Public Attributes

- static constexpr int `digits`
- static constexpr int `digits10`
- static constexpr `float_denorm_style` `has_denorm`
- static constexpr bool `has_denorm_loss`
- static constexpr bool `has_infinity`
- static constexpr bool `has_quiet_NaN`
- static constexpr bool `has_signaling_NaN`
- static constexpr bool `is_bounded`
- static constexpr bool `is_exact`
- static constexpr bool `is_iec559`
- static constexpr bool `is_integer`
- static constexpr bool `is_modulo`
- static constexpr bool `is_signed`
- static constexpr bool `is_specialized`
- static constexpr int `max_digits10`
- static constexpr int `max_exponent`
- static constexpr int `max_exponent10`
- static constexpr int `min_exponent`
- static constexpr int `min_exponent10`
- static constexpr int `radix`
- static constexpr `float_round_style` `round_style`
- static constexpr bool `tinyness_before`
- static constexpr bool `traps`

#### 4.756.1 Detailed Description

```
template<typename _Tp>
struct std::numeric_limits<_Tp>
```

Properties of fundamental types.

This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

Definition at line 312 of file `limits`.

## 4.756.2 Member Function Documentation

## 4.756.2.1 denorm\_min()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::denorm_min () [inline], [static], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is `false`, this is the minimum positive normalized value.

Definition at line 357 of file `limits`.

## 4.756.2.2 epsilon()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::epsilon () [inline], [static], [noexcept]
```

The *machine epsilon*: the difference between 1 and the least value greater than 1 that is representable.

Definition at line 333 of file `limits`.

Referenced by `std::generate_canonical()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

## 4.756.2.3 infinity()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::infinity () [inline], [static], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

Definition at line 341 of file `limits`.

## 4.756.2.4 lowest()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::lowest () [inline], [static], [noexcept]
```

A finite value `x` such that there is no other finite value `y` where `y < x`.

Definition at line 327 of file `limits`.

Referenced by `std::normal_distribution< result_type >::min()`, `std::cauchy_distribution< _RealType >::min()`, `std::student_t_distribution< _RealType >::min()`, and `std::extreme_value_distribution< _RealType >::min()`.

#### 4.756.2.5 max()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::max () [inline], [static], [noexcept]
```

The maximum finite value.

Definition at line 321 of file limits.

Referenced by `std::normal_distribution< result_type >::max()`, `std::lognormal_distribution< _RealType >::max()`, `std::gamma_distribution< result_type >::max()`, `std::chi_squared_distribution< _RealType >::max()`, `std::cauchy_distribution< _RealType >::max()`, `std::fisher_f_distribution< _RealType >::max()`, `std::student_t_distribution< _RealType >::max()`, `std::bernoulli_distribution::max()`, `std::geometric_distribution< _IntType >::max()`, `std::negative_binomial_distribution< _IntType >::max()`, `std::poisson_distribution< _IntType >::max()`, `std::exponential_distribution< _RealType >::max()`, `std::weibull_distribution< _RealType >::max()`, `std::extreme_value_distribution< _RealType >::max()`, `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()()`, `std::binomial_distribution< _IntType >::operator()()`, and `std::poisson_distribution< _IntType >::operator()()`.

#### 4.756.2.6 min()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::min () [inline], [static], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Definition at line 317 of file limits.

Referenced by `std::bernoulli_distribution::min()`, and `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()()`.

#### 4.756.2.7 quiet\_NaN()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN () [inline], [static], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

Definition at line 346 of file limits.

#### 4.756.2.8 round\_error()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits< _Tp >::round_error () [inline], [static], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

Definition at line 337 of file limits.

#### 4.756.2.9 signaling\_NaN()

```
template<typename _Tp >
static constexpr _Tp std::numeric_limits<_Tp>::signaling_NaN () [inline], [static], [noexcept]
```

The representation of a signaling Not a Number, if has\_signaling\_NaN.

Definition at line 351 of file limits.

### 4.756.3 Member Data Documentation

#### 4.756.3.1 digits

```
constexpr int std::__numeric_limits_base::digits [static], [inherited]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

Definition at line 211 of file limits.

#### 4.756.3.2 digits10

```
constexpr int std::__numeric_limits_base::digits10 [static], [inherited]
```

The number of base 10 digits that can be represented without change.

Definition at line 214 of file limits.

#### 4.756.3.3 has\_denorm

```
constexpr float_denorm_style std::__numeric_limits_base::has_denorm [static], [inherited]
```

See `std::float_denorm_style` for more information.

Definition at line 266 of file limits.

#### 4.756.3.4 has\_denorm\_loss

```
constexpr bool std::__numeric_limits_base::has_denorm_loss [static], [inherited]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

Definition at line 270 of file limits.

#### 4.756.3.5 has\_infinity

```
constexpr bool std::__numeric_limits_base::has_infinity [static], [inherited]
```

True if the type has a representation for positive infinity.

Definition at line 255 of file limits.

#### 4.756.3.6 has\_quiet\_NaN

```
constexpr bool std::__numeric_limits_base::has_quiet_NaN [static], [inherited]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

Definition at line 259 of file limits.

#### 4.756.3.7 has\_signaling\_NaN

```
constexpr bool std::__numeric_limits_base::has_signaling_NaN [static], [inherited]
```

True if the type has a representation for a signaling Not a Number.

Definition at line 263 of file limits.

#### 4.756.3.8 is\_bounded

```
constexpr bool std::__numeric_limits_base::is_bounded [static], [inherited]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

Definition at line 279 of file limits.

#### 4.756.3.9 is\_exact

```
constexpr bool std::__numeric_limits_base::is_exact [static], [inherited]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

Definition at line 231 of file limits.

#### 4.756.3.10 is\_iec559

```
constexpr bool std::__numeric_limits_base::is_iec559 [static], [inherited]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

Definition at line 274 of file limits.

#### 4.756.3.11 is\_integer

```
constexpr bool std::__numeric_limits_base::is_integer [static], [inherited]
```

True if the type is integer.

Definition at line 226 of file limits.

#### 4.756.3.12 is\_modulo

```
constexpr bool std::__numeric_limits_base::is_modulo [static], [inherited]
```

True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or \* on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

Definition at line 288 of file limits.

#### 4.756.3.13 is\_signed

```
constexpr bool std::__numeric_limits_base::is_signed [static], [inherited]
```

True if the type is signed.

Definition at line 223 of file limits.

#### 4.756.3.14 is\_specialized

```
constexpr bool std::__numeric_limits_base::is_specialized [static], [inherited]
```

This will be true for all fundamental types (which have specializations), and false for everything else.

Definition at line 206 of file limits.

**4.756.3.15 max\_digits10**

```
constexpr int std::__numeric_limits_base::max_digits10 [static], [inherited]
```

The number of base 10 digits required to ensure that values which differ are always differentiated.

Definition at line 219 of file limits.

**4.756.3.16 max\_exponent**

```
constexpr int std::__numeric_limits_base::max_exponent [static], [inherited]
```

The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

Definition at line 248 of file limits.

**4.756.3.17 max\_exponent10**

```
constexpr int std::__numeric_limits_base::max_exponent10 [static], [inherited]
```

The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

Definition at line 252 of file limits.

**4.756.3.18 min\_exponent**

```
constexpr int std::__numeric_limits_base::min_exponent [static], [inherited]
```

The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

Definition at line 239 of file limits.

**4.756.3.19 min\_exponent10**

```
constexpr int std::__numeric_limits_base::min_exponent10 [static], [inherited]
```

The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

Definition at line 243 of file limits.



#### 4.756.3.20 radix

```
constexpr int std::__numeric_limits_base::radix [static], [inherited]
```

For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Definition at line 235 of file limits.

#### 4.756.3.21 round\_style

```
constexpr float_round_style std::__numeric_limits_base::round_style [static], [inherited]
```

See std::float\_round\_style for more information. This is only meaningful for floating types; integer types will all be round\_toward\_zero.

Definition at line 299 of file limits.

#### 4.756.3.22 tinyness\_before

```
constexpr bool std::__numeric_limits_base::tinyness_before [static], [inherited]
```

True if tininess is detected before rounding. (see IEC 559)

Definition at line 294 of file limits.

#### 4.756.3.23 traps

```
constexpr bool std::__numeric_limits_base::traps [static], [inherited]
```

True if trapping is implemented for this type.

Definition at line 291 of file limits.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.757 `std::numeric_limits< bool >` Struct Template Reference

### Static Public Member Functions

- static constexpr bool **denorm\_min** () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool **min** () noexcept
- static constexpr bool **quiet\_NaN** () noexcept
- static constexpr bool **round\_error** () noexcept
- static constexpr bool **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.757.1 Detailed Description

```
template<>
struct std::numeric_limits< bool >
```

`numeric_limits<bool>` specialization.

Definition at line 384 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.758 `std::numeric_limits< char >` Struct Template Reference

### Static Public Member Functions

- static constexpr char **denorm\_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet\_NaN** () noexcept
- static constexpr char **round\_error** () noexcept
- static constexpr char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.758.1 Detailed Description

```
template<>
struct std::numeric_limits< char >
```

`numeric_limits<char>` specialization.

Definition at line 453 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.759 `std::numeric_limits< char16_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr char16\_t **denorm\_min** () noexcept
- static constexpr char16\_t **epsilon** () noexcept
- static constexpr char16\_t **infinity** () noexcept
- static constexpr char16\_t **lowest** () noexcept
- static constexpr char16\_t **max** () noexcept
- static constexpr char16\_t **min** () noexcept
- static constexpr char16\_t **quiet\_NaN** () noexcept
- static constexpr char16\_t **round\_error** () noexcept
- static constexpr char16\_t **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr float **denorm\_style** **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr float **round\_style** **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.759.1 Detailed Description

```
template<>
struct std::numeric_limits< char16_t >
```

`numeric_limits<char16_t>` specialization.

Definition at line 797 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.760 `std::numeric_limits< char32_t >` Struct Template Reference

## Static Public Member Functions

- static constexpr `char32_t` **denorm\_min** () noexcept
- static constexpr `char32_t` **epsilon** () noexcept
- static constexpr `char32_t` **infinity** () noexcept
- static constexpr `char32_t` **lowest** () noexcept
- static constexpr `char32_t` **max** () noexcept
- static constexpr `char32_t` **min** () noexcept
- static constexpr `char32_t` **quiet\_NaN** () noexcept
- static constexpr `char32_t` **round\_error** () noexcept
- static constexpr `char32_t` **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr `float_denorm_style` **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr `float_round_style` **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.760.1 Detailed Description

```
template<>
struct std::numeric_limits< char32_t >
```

`numeric_limits<char32_t>` specialization.

Definition at line 858 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.761 `std::numeric_limits< double >` Struct Template Reference

### Static Public Member Functions

- static constexpr double **denorm\_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet\_NaN** () noexcept
- static constexpr double **round\_error** () noexcept
- static constexpr double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.761.1 Detailed Description

```
template<>
struct std::numeric_limits< double >
```

`numeric_limits<double>` specialization.

Definition at line 1739 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.762 `std::numeric_limits< float >` Struct Template Reference

## Static Public Member Functions

- static constexpr float **denorm\_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet\_NaN** () noexcept
- static constexpr float **round\_error** () noexcept
- static constexpr float **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.762.1 Detailed Description

```
template<>
struct std::numeric_limits< float >
```

`numeric_limits<float>` specialization.

Definition at line 1664 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.763 `std::numeric_limits< int >` Struct Template Reference

### Static Public Member Functions

- static constexpr int **denorm\_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet\_NaN** () noexcept
- static constexpr int **round\_error** () noexcept
- static constexpr int **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.763.1 Detailed Description

```
template<>
struct std::numeric_limits< int >
```

`numeric_limits<int>` specialization.

Definition at line 1060 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)



4.764 `std::numeric_limits< long >` Struct Template Reference

## Static Public Member Functions

- static constexpr long **denorm\_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet\_NaN** () noexcept
- static constexpr long **round\_error** () noexcept
- static constexpr long **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.764.1 Detailed Description

```
template<>
struct std::numeric_limits< long >
```

`numeric_limits<long>` specialization.

Definition at line 1199 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.765 `std::numeric_limits< long double >` Struct Template Reference

### Static Public Member Functions

- static constexpr long double **denorm\_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet\_NaN** () noexcept
- static constexpr long double **round\_error** () noexcept
- static constexpr long double **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.765.1 Detailed Description

```
template<>
struct std::numeric_limits< long double >
```

`numeric_limits<long double>` specialization.

Definition at line 1814 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.766 `std::numeric_limits< long long >` Struct Template Reference

### Static Public Member Functions

- static constexpr long long **denorm\_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet\_NaN** () noexcept
- static constexpr long long **round\_error** () noexcept
- static constexpr long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.766.1 Detailed Description

```
template<>
struct std::numeric_limits< long long >
```

`numeric_limits<long long>` specialization.

Definition at line 1339 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.767 `std::numeric_limits< short >` Struct Template Reference

### Static Public Member Functions

- static constexpr short **denorm\_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet\_NaN** () noexcept
- static constexpr short **round\_error** () noexcept
- static constexpr short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.767.1 Detailed Description

```
template<>
struct std::numeric_limits< short >
```

`numeric_limits<short>` specialization.

Definition at line 920 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.768 `std::numeric_limits< signed char >` Struct Template Reference

## Static Public Member Functions

- static constexpr signed char **denorm\_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet\_NaN** () noexcept
- static constexpr signed char **round\_error** () noexcept
- static constexpr signed char **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.768.1 Detailed Description

```
template<>
struct std::numeric_limits< signed char >
```

`numeric_limits<signed char>` specialization.

Definition at line 520 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.769 `std::numeric_limits< unsigned char >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned char **denorm\_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet\_NaN** () noexcept
- static constexpr unsigned char **round\_error** () noexcept
- static constexpr unsigned char **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.769.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned char >
```

`numeric_limits<unsigned char>` specialization.

Definition at line 590 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

4.770 `std::numeric_limits< unsigned int >` Struct Template Reference

## Static Public Member Functions

- static constexpr unsigned int **denorm\_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet\_NaN** () noexcept
- static constexpr unsigned int **round\_error** () noexcept
- static constexpr unsigned int **signaling\_NaN** () noexcept

## Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

## 4.770.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned int >
```

`numeric_limits<unsigned int>` specialization.

Definition at line 1127 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.771 `std::numeric_limits< unsigned long >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned long **denorm\_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet\_NaN** () noexcept
- static constexpr unsigned long **round\_error** () noexcept
- static constexpr unsigned long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.771.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned long >
```

`numeric_limits<unsigned long>` specialization.

Definition at line 1266 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)



## 4.772 `std::numeric_limits< unsigned long long >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned long long **denorm\_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet\_NaN** () noexcept
- static constexpr unsigned long long **round\_error** () noexcept
- static constexpr unsigned long long **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.772.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned long long >
```

`numeric_limits<unsigned long long>` specialization.

Definition at line 1409 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.773 `std::numeric_limits< unsigned short >` Struct Template Reference

### Static Public Member Functions

- static constexpr unsigned short **denorm\_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet\_NaN** () noexcept
- static constexpr unsigned short **round\_error** () noexcept
- static constexpr unsigned short **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr [float\\_denorm\\_style](#) **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr [float\\_round\\_style](#) **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.773.1 Detailed Description

```
template<>
struct std::numeric_limits< unsigned short >
```

`numeric_limits<unsigned short>` specialization.

Definition at line 987 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.774 `std::numeric_limits< wchar_t >` Struct Template Reference

### Static Public Member Functions

- static constexpr `wchar_t` **denorm\_min** () noexcept
- static constexpr `wchar_t` **epsilon** () noexcept
- static constexpr `wchar_t` **infinity** () noexcept
- static constexpr `wchar_t` **lowest** () noexcept
- static constexpr `wchar_t` **max** () noexcept
- static constexpr `wchar_t` **min** () noexcept
- static constexpr `wchar_t` **quiet\_NaN** () noexcept
- static constexpr `wchar_t` **round\_error** () noexcept
- static constexpr `wchar_t` **signaling\_NaN** () noexcept

### Static Public Attributes

- static constexpr int **digits**
- static constexpr int **digits10**
- static constexpr `float_denorm_style` **has\_denorm**
- static constexpr bool **has\_denorm\_loss**
- static constexpr bool **has\_infinity**
- static constexpr bool **has\_quiet\_NaN**
- static constexpr bool **has\_signaling\_NaN**
- static constexpr bool **is\_bounded**
- static constexpr bool **is\_exact**
- static constexpr bool **is\_iec559**
- static constexpr bool **is\_integer**
- static constexpr bool **is\_modulo**
- static constexpr bool **is\_signed**
- static constexpr bool **is\_specialized**
- static constexpr int **max\_digits10**
- static constexpr int **max\_exponent**
- static constexpr int **max\_exponent10**
- static constexpr int **min\_exponent**
- static constexpr int **min\_exponent10**
- static constexpr int **radix**
- static constexpr `float_round_style` **round\_style**
- static constexpr bool **tinyness\_before**
- static constexpr bool **traps**

### 4.774.1 Detailed Description

```
template<>
struct std::numeric_limits< wchar_t >
```

`numeric_limits<wchar_t>` specialization.

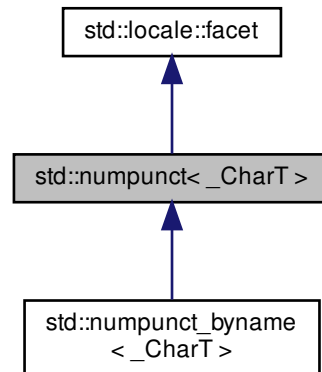
Definition at line 663 of file `limits`.

The documentation for this struct was generated from the following file:

- [limits](#)

## 4.775 std::num\_punct<\_CharT> Class Template Reference

Inheritance diagram for std::num\_punct<\_CharT>:



### Public Types

- typedef `__num_punct_cache<_CharT>` **`__cache_type`**
- typedef `_CharT` `char_type`
- typedef `basic_string<_CharT>` `string_type`

### Public Member Functions

- `num_punct` (`size_t __refs=0`)
- `num_punct` (`__cache_type *__cache, size_t __refs=0`)
- `num_punct` (`__c_locale __cloc, size_t __refs=0`)
- `char_type decimal_point` () const
- `string_type falsename` () const
- `string_type grouping` () const
- `char_type thousands_sep` () const
- `string_type truename` () const

### Static Public Attributes

- static `locale::id` `id`

## Protected Member Functions

- virtual [~numpunct](#) ()
- void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc=0)
- template<>  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- template<>  
void [\\_M\\_initialize\\_numpunct](#) (\_\_c\_locale \_\_cloc)
- virtual [char\\_type do\\_decimal\\_point](#) () const
- virtual [string\\_type do\\_falsename](#) () const
- virtual [string do\\_grouping](#) () const
- virtual [char\\_type do\\_thousands\\_sep](#) () const
- virtual [string\\_type do\\_truename](#) () const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## Protected Attributes

- [\\_\\_cache\\_type](#) \* [\\_M\\_data](#)

## 4.775.1 Detailed Description

```
template<typename _CharT>
class std::numpunct<_CharT>
```

Primary class template numpunct.

This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.

The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

Definition at line 1670 of file locale\_facets.h.

## 4.775.2 Member Typedef Documentation

#### 4.775.2.1 char\_type

```
template<typename _CharT >
typedef _CharT std::num_punct< _CharT >::char_type
```

Public typedefs.

Definition at line 1676 of file locale\_facets.h.

#### 4.775.2.2 string\_type

```
template<typename _CharT >
typedef basic_string<_CharT> std::num_punct< _CharT >::string_type
```

Public typedefs.

Definition at line 1677 of file locale\_facets.h.

### 4.775.3 Constructor & Destructor Documentation

#### 4.775.3.1 num\_punct() [1/3]

```
template<typename _CharT >
std::num_punct< _CharT >::num_punct (
 size_t __refs = 0) [inline], [explicit]
```

Num\_punct constructor.

Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__refs</code> | Refcount to pass to the base class. |
|---------------------|-------------------------------------|

Definition at line 1694 of file locale\_facets.h.

#### 4.775.3.2 num\_punct() [2/3]

```
template<typename _CharT >
std::num_punct< _CharT >::num_punct (
 __cache_type * __cache,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up the predefined locale facets.

## Parameters

|                      |                                     |
|----------------------|-------------------------------------|
| <code>__cache</code> | __numpunct_cache object.            |
| <code>__refs</code>  | Refcount to pass to the base class. |

Definition at line 1708 of file locale\_facets.h.

## 4.775.3.3 numpunct() [3/3]

```
template<typename _CharT >
std::numpunct< _CharT >::numpunct (
 __c_locale __cloc,
 size_t __refs = 0) [inline], [explicit]
```

Internal constructor. Not for general use.

This is a constructor for use by the library itself to set up new locales.

## Parameters

|                     |                                     |
|---------------------|-------------------------------------|
| <code>__cloc</code> | The C locale.                       |
| <code>__refs</code> | Refcount to pass to the base class. |

Definition at line 1722 of file locale\_facets.h.

## 4.775.3.4 ~numpunct()

```
template<typename _CharT >
virtual std::numpunct< _CharT >::~numpunct () [protected], [virtual]
```

Destructor.

## 4.775.4 Member Function Documentation

## 4.775.4.1 decimal\_point()

```
template<typename _CharT >
char_type std::numpunct< _CharT >::decimal_point () const [inline]
```

Return decimal point character.

This function returns a char\_type to use as a decimal point. It does so by returning returning numpunct<char\_type>↔::do\_decimal\_point().

**Returns**

*char\_type* representing a decimal point.

Definition at line 1736 of file locale\_facets.h.

References `std::num_punct<_CharT>::do_decimal_point()`.

**4.775.4.2 do\_decimal\_point()**

```
template<typename _CharT >
virtual char_type std::num_punct<_CharT>::do_decimal_point () const [inline], [protected], [virtual]
```

Return decimal point character.

Returns a *char\_type* to use as a decimal point. This function is a hook for derived classes to change the value returned.

**Returns**

*char\_type* representing a decimal point.

Definition at line 1823 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::decimal_point()`.

**4.775.4.3 do\_falsename()**

```
template<typename _CharT >
virtual string_type std::num_punct<_CharT>::do_falsename () const [inline], [protected], [virtual]
```

Return string representation of bool false.

Returns a *string\_type* containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

**Returns**

*string\_type* representing printed form of false.

Definition at line 1874 of file locale\_facets.h.

Referenced by `std::num_punct<_CharT>::falsename()`.



#### 4.775.4.4 do\_grouping()

```
template<typename _CharT >
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1848 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::grouping().

#### 4.775.4.5 do\_thousands\_sep()

```
template<typename _CharT >
virtual char_type std::num_punct<_CharT>::do_thousands_sep () const [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char\_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a thousands separator.

Definition at line 1835 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::thousands\_sep().

#### 4.775.4.6 do\_truename()

```
template<typename _CharT >
virtual string_type std::num_punct< _CharT >::do_truename () const [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string\_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

##### Returns

string\_type representing printed form of true.

Definition at line 1861 of file locale\_facets.h.

Referenced by std::num\_punct< \_CharT >::truename().

#### 4.775.4.7 falsename()

```
template<typename _CharT >
string_type std::num_punct< _CharT >::falsename () const [inline]
```

Return string representation of bool false.

This function returns a string\_type containing the text representation for false bool variables. It does so by calling num\_punct<char\_type>::do\_falsename().

##### Returns

string\_type representing printed form of false.

Definition at line 1806 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_falsename().

## 4.775.4.8 grouping()

```
template<typename _CharT >
string std::num_punct<_CharT>::grouping () const [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

**Returns**

string representing grouping specification.

Definition at line 1780 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_grouping().

## 4.775.4.9 thousands\_sep()

```
template<typename _CharT >
char_type std::num_punct<_CharT>::thousands_sep () const [inline]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

**Returns**

char\_type representing a thousands separator.

Definition at line 1749 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_thousands\_sep().

#### 4.775.4.10 true\_name()

```
template<typename _CharT >
string_type std::num_punct< _CharT >::true_name () const [inline]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling num\_punct<char\_type>::do\_true\_name().

##### Returns

string\_type representing printed form of true.

Definition at line 1793 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_true\_name().

### 4.775.5 Member Data Documentation

#### 4.775.5.1 id

```
template<typename _CharT >
locale::id std::num_punct< _CharT >::id [static]
```

Num\_punct facet id.

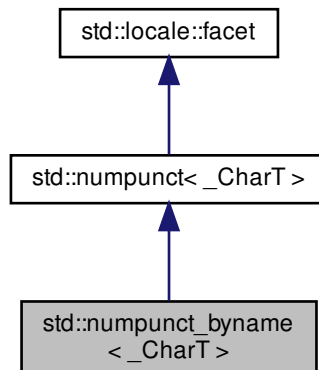
Definition at line 1686 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

### 4.776 std::num\_punct\_byname< \_CharT > Class Template Reference

Inheritance diagram for std::num\_punct\_byname< \_CharT >:



### Public Types

- typedef \_\_num\_punct\_cache<\_CharT> **\_\_cache\_type**
- typedef \_CharT **char\_type**
- typedef basic\_string<\_CharT> **string\_type**

### Public Member Functions

- **num\_punct\_byname** (const char \*\_\_s, size\_t \_\_refs=0)
- **num\_punct\_byname** (const string &\_\_s, size\_t \_\_refs=0)
- **char\_type decimal\_point** () const
- **string\_type falsename** () const
- **string grouping** () const
- **char\_type thousands\_sep** () const
- **string\_type truename** () const

### Static Public Attributes

- static locale::id id

### Protected Member Functions

- void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc=0)
- template<>  
void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- template<>  
void **\_M\_initialize\_numpunct** (\_\_c\_locale \_\_cloc)
- virtual **char\_type do\_decimal\_point** () const
- virtual **string\_type do\_falsename** () const
- virtual **string do\_grouping** () const
- virtual **char\_type do\_thousands\_sep** () const
- virtual **string\_type do\_truename** () const

### Static Protected Member Functions

- static \_\_c\_locale **\_S\_clone\_c\_locale** (\_\_c\_locale &\_\_cloc) throw ()
- static void **\_S\_create\_c\_locale** (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void **\_S\_destroy\_c\_locale** (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale **\_S\_get\_c\_locale** ()
- static const char \* **\_S\_get\_c\_name** () throw ()
- static \_\_c\_locale **\_S\_lc\_ctype\_c\_locale** (\_\_c\_locale \_\_cloc, const char \*\_\_s)

### Protected Attributes

- \_\_cache\_type \* **\_M\_data**

#### 4.776.1 Detailed Description

```
template<typename _CharT>
class std::numpunct_byname<_CharT>
```

class numpunct\_byname [22.2.3.2].

Definition at line 1903 of file locale\_facets.h.

#### 4.776.2 Member Function Documentation

##### 4.776.2.1 decimal\_point()

```
template<typename _CharT >
char_type std::numpunct<_CharT>::decimal_point () const [inline], [inherited]
```

Return decimal point character.

This function returns a `char_type` to use as a decimal point. It does so by returning `returning numpunct<char_type>::do_decimal_point()`.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1736 of file locale\_facets.h.

References `std::numpunct<_CharT>::do_decimal_point()`.

##### 4.776.2.2 do\_decimal\_point()

```
template<typename _CharT >
virtual char_type std::numpunct<_CharT>::do_decimal_point () const [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.

Returns a `char_type` to use as a decimal point. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a decimal point.

Definition at line 1823 of file locale\_facets.h.

Referenced by `std::numpunct<_CharT>::decimal_point()`.

#### 4.776.2.3 do\_falsename()

```
template<typename _CharT >
virtual string_type std::num_punct<_CharT>::do_falsename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.

Returns a string\_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

##### Returns

string\_type representing printed form of false.

Definition at line 1874 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::falsename().

#### 4.776.2.4 do\_grouping()

```
template<typename _CharT >
virtual string std::num_punct<_CharT>::do_grouping () const [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

##### See also

grouping() for details.

##### Returns

String representing grouping specification.

Definition at line 1848 of file locale\_facets.h.

Referenced by std::num\_punct<\_CharT>::grouping().

#### 4.776.2.5 do\_thousands\_sep()

```
template<typename _CharT >
virtual char_type std::num_punct< _CharT >::do_thousands_sep () const [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a `char_type` to use as a thousands separator. This function is a hook for derived classes to change the value returned.

##### Returns

*char\_type* representing a thousands separator.

Definition at line 1835 of file `locale_facets.h`.

Referenced by `std::num_punct< _CharT >::thousands_sep()`.

#### 4.776.2.6 do\_truename()

```
template<typename _CharT >
virtual string_type std::num_punct< _CharT >::do_truename () const [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.

Returns a `string_type` containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

##### Returns

`string_type` representing printed form of true.

Definition at line 1861 of file `locale_facets.h`.

Referenced by `std::num_punct< _CharT >::truename()`.

#### 4.776.2.7 falsename()

```
template<typename _CharT >
string_type std::num_punct< _CharT >::falsename () const [inline], [inherited]
```

Return string representation of bool false.

This function returns a `string_type` containing the text representation for false bool variables. It does so by calling `num_punct<char_type>::do_falsename()`.

##### Returns

`string_type` representing printed form of false.

Definition at line 1806 of file `locale_facets.h`.

References `std::num_punct< _CharT >::do_falsename()`.



#### 4.776.2.8 grouping()

```
template<typename _CharT >
string std::num_punct<_CharT>::grouping () const [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpreted as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling num\_punct<char\_type>::do\_grouping().

##### Returns

string representing grouping specification.

Definition at line 1780 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_grouping().

#### 4.776.2.9 thousands\_sep()

```
template<typename _CharT >
char_type std::num_punct<_CharT>::thousands_sep () const [inline], [inherited]
```

Return thousands separator character.

This function returns a char\_type to use as a thousands separator. It does so by returning num\_punct<char\_type>::do\_thousands\_sep().

##### Returns

char\_type representing a thousands separator.

Definition at line 1749 of file locale\_facets.h.

References std::num\_punct<\_CharT>::do\_thousands\_sep().

#### 4.776.2.10 truename()

```
template<typename _CharT >
string_type std::num_punct< _CharT >::truename () const [inline], [inherited]
```

Return string representation of bool true.

This function returns a string\_type containing the text representation for true bool variables. It does so by calling num\_punct<char\_type>::do\_truename().

##### Returns

string\_type representing printed form of true.

Definition at line 1793 of file locale\_facets.h.

References std::num\_punct< \_CharT >::do\_truename().

### 4.776.3 Member Data Documentation

#### 4.776.3.1 id

```
template<typename _CharT >
locale::id std::num_punct< _CharT >::id [static], [inherited]
```

Num\_punct facet id.

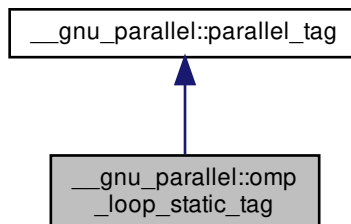
Definition at line 1686 of file locale\_facets.h.

The documentation for this class was generated from the following file:

- [locale\\_facets.h](#)

### 4.777 \_\_gnu\_parallel::omp\_loop\_static\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_static\_tag:



## Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads\(\)](#)
- void [set\\_num\\_threads\(\\_ThreadIndex \\_\\_num\\_threads\)](#)

## 4.777.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

Definition at line 100 of file tags.h.

## 4.777.2 Member Function Documentation

4.777.2.1 `__get_num_threads()`

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

## Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

4.777.2.2 `set_num_threads()`

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

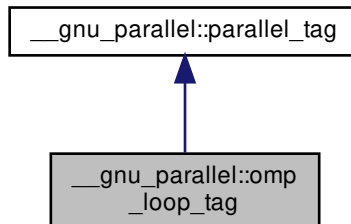
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.778 \_\_gnu\_parallel::omp\_loop\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::omp\_loop\_tag:



##### Public Member Functions

- [\\_ThreadIndex \\_\\_get\\_num\\_threads \( \)](#)
- void [set\\_num\\_threads \( \\_ThreadIndex \\_\\_num\\_threads \)](#)

##### 4.778.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

Definition at line 96 of file tags.h.

##### 4.778.2 Member Function Documentation

###### 4.778.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

###### 4.778.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.779 std::once\_flag Struct Reference

## Public Member Functions

- constexpr [once\\_flag](#) () noexcept=default
- [once\\_flag](#) (const [once\\_flag](#) &)=delete
- [once\\_flag](#) & [operator=](#) (const [once\\_flag](#) &)=delete

## Friends

- template<typename \_Callable , typename... \_Args>  
void [call\\_once](#) ([once\\_flag](#) &\_\_once, \_Callable &&\_\_f, \_Args &&... \_\_args)

## 4.779.1 Detailed Description

Flag type used by std::call\_once.

Definition at line 672 of file mutex.

## 4.779.2 Constructor &amp; Destructor Documentation

## 4.779.2.1 once\_flag() [1/2]

```
constexpr std::once_flag::once_flag () [default], [noexcept]
```

Constructor.

#### 4.779.2.2 `once_flag()` [2/2]

```
std::once_flag::once_flag (
 const once_flag &) [delete]
```

Deleted copy constructor.

#### 4.779.3 Member Function Documentation

##### 4.779.3.1 `operator=()`

```
once_flag& std::once_flag::operator= (
 const once_flag &) [delete]
```

Deleted assignment operator.

#### 4.779.4 Friends And Related Function Documentation

##### 4.779.4.1 `call_once`

```
template<typename _Callable , typename... _Args>
void call_once (
 once_flag & __once,
 _Callable && __f,
 _Args &&... __args) [friend]
```

Invoke a callable and synchronize with other calls using the same flag.

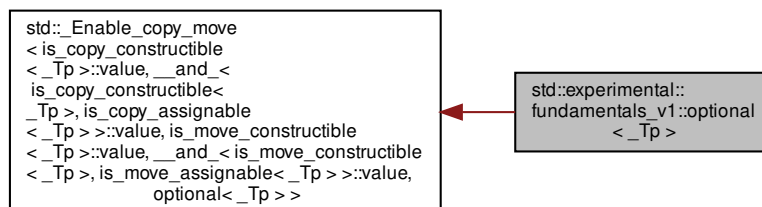
Definition at line 712 of file `mutex`.

The documentation for this struct was generated from the following file:

- [mutex](#)

#### 4.780 `std::experimental::fundamentals_v1::optional<_Tp>` Class Template Reference

Inheritance diagram for `std::experimental::fundamentals_v1::optional<_Tp>`:



## Public Types

- using **value\_type** = \_Tp

## Public Member Functions

- template<typename \_Up = \_Tp, enable\_if\_t< \_\_and< \_\_not< is\_same< optional< \_Tp >, decay\_t< \_Up >>>, is\_constructible< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp >>::value, bool > = true>  
constexpr **optional** (\_Up &&\_\_t)
- template<typename \_Up = \_Tp, enable\_if\_t< \_\_and< \_\_not< is\_same< optional< \_Tp >, decay\_t< \_Up >>>, is\_constructible< \_Tp, \_Up && >, \_\_not< is\_convertible< \_Up &&, \_Tp >>::value, bool > = false>  
constexpr **optional** (\_Up &&\_\_t)
- template<typename \_Up, enable\_if\_t< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, const \_Up & >, is\_convertible< const \_Up &, \_Tp >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>::value, bool > = true>  
constexpr **optional** (const **optional**< \_Up > &\_\_t)
- template<typename \_Up, enable\_if\_t< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, const \_Up & >, \_\_not< is\_convertible< const \_Up &, \_Tp >>, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>::value, bool > = false>  
constexpr **optional** (const **optional**< \_Up > &\_\_t)
- template<typename \_Up, enable\_if\_t< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, \_Up && >, is\_convertible< \_Up &&, \_Tp >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>::value, bool > = true>  
constexpr **optional** (**optional**< \_Up > &&\_\_t)
- template<typename \_Up, enable\_if\_t< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, \_Up && >, \_\_not< is\_convertible< \_Up &&, \_Tp >>, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>::value, bool > = false>  
constexpr **optional** (**optional**< \_Up > &&\_\_t)
- template<typename... \_Args>  
**enable\_if\_t**< **is\_constructible**< \_Tp, \_Args &&... >::value > **emplace** (\_Args &&... \_\_args)
- template<typename \_Up, typename... \_Args>  
**enable\_if\_t**< **is\_constructible**< \_Tp, **initializer\_list**< \_Up > &, \_Args &&... >::value > **emplace** (**initializer\_list**< \_Up > \_\_il, \_Args &&... \_\_args)
- constexpr **operator bool** () const noexcept
- constexpr const \_Tp & **operator\*** () const &
- constexpr \_Tp & **operator\*** () &
- constexpr \_Tp && **operator\*** () &&
- constexpr const \_Tp && **operator\*** () const &&
- constexpr const \_Tp \* **operator->** () const
- \_Tp \* **operator->** ()
- **optional** & **operator=** (nullopt\_t) noexcept
- template<typename \_Up = \_Tp>  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< optional< \_Tp >, decay\_t< \_Up >>>, is\_constructible< \_Tp, \_Up >, \_\_not< \_\_and< is\_scalar< \_Tp >, is\_same< \_Tp, decay\_t< \_Up >>>, is\_assignable< \_Tp &, \_Up >>::value, **optional** & > **operator=** (\_Up &&\_\_u)
- template<typename \_Up >  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, const \_Up & >, is\_assignable< \_Tp &, \_Up >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>, \_\_not< \_\_assigns\_from\_optional< \_Tp, \_Up >>::value, **optional** & > **operator=** (const **optional**< \_Up > &\_\_u)
- template<typename \_Up >  
**enable\_if\_t**< \_\_and< \_\_not< is\_same< \_Tp, \_Up >>, is\_constructible< \_Tp, \_Up >, is\_assignable< \_Tp &, \_Up >, \_\_not< \_\_converts\_from\_optional< \_Tp, \_Up >>, \_\_not< \_\_assigns\_from\_optional< \_Tp, \_Up >>::value, **optional** & > **operator=** (**optional**< \_Up > &&\_\_u)
- void **swap** (**optional** &\_\_other) noexcept(**is\_nothrow\_move\_constructible**< \_Tp >()) &&\_\_is\_nothrow\_swappable< \_Tp >::value)
- constexpr const \_Tp & **value** () const &

- constexpr `_Tp & value ()` &
- constexpr `_Tp && value () &&`
- constexpr `const _Tp && value () const &&`
- template<typename `_Up` >  
constexpr `_Tp value_or (_Up &&__u) const &`
- template<typename `_Up` >  
`_Tp value_or (_Up &&__u) &&`

#### 4.780.1 Detailed Description

```
template<typename _Tp>
class std::experimental::fundamentals_v1::optional<_Tp>
```

Class template for optional values.

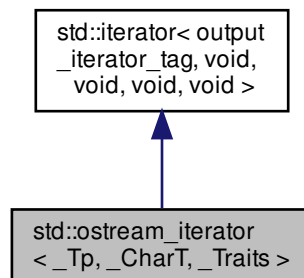
Definition at line 70 of file experimental/optional.

The documentation for this class was generated from the following file:

- [experimental/optional](#)

#### 4.781 `std::ostream_iterator<_Tp, _CharT, _Traits>` Class Template Reference

Inheritance diagram for `std::ostream_iterator<_Tp, _CharT, _Traits>`:





## Public Types

- typedef void [difference\\_type](#)
  - typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
  - typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- 
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

## Public Member Functions

- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s)
- [ostream\\_iterator](#) ([ostream\\_type](#) &\_\_s, const [\\_CharT](#) \*\_\_c)
- [ostream\\_iterator](#) (const [ostream\\_iterator](#) &\_\_obj)
- [ostream\\_iterator](#) & [operator\\*](#) ()
- [ostream\\_iterator](#) & [operator++](#) ()
- [ostream\\_iterator](#) & [operator++](#) (int)
- [ostream\\_iterator](#) & [operator=](#) (const [ostream\\_iterator](#) &)=default
- [ostream\\_iterator](#) & [operator=](#) (const [\\_Tp](#) &\_\_value)

## 4.781.1 Detailed Description

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >
```

Provides output iterator semantics for streams.

This class provides an iterator to write to an ostream. The type `Tp` is the only type written by this iterator and there must be an `operator<<(Tp)` defined.

## Template Parameters

|                      |                                        |
|----------------------|----------------------------------------|
| <code>_Tp</code>     | The type to write to the ostream.      |
| <code>_CharT</code>  | The ostream <code>char_type</code> .   |
| <code>_Traits</code> | The ostream <code>char_traits</code> . |

Definition at line 176 of file `stream_iterator.h`.

## 4.781.2 Member Typedef Documentation

#### 4.781.2.1 char\_type

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type
```

Public typedef.

Definition at line 185 of file stream\_iterator.h.

#### 4.781.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file stl\_iterator\_base\_types.h.

#### 4.781.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >←
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.781.2.4 ostream\_type

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_type
```

Public typedef.

Definition at line 187 of file stream\_iterator.h.

#### 4.781.2.5 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file stl\_iterator\_base\_types.h.

## 4.781.2.6 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file stl\_iterator\_base\_types.h.

## 4.781.2.7 traits\_type

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
```

Public typedef.

Definition at line 186 of file stream\_iterator.h.

## 4.781.2.8 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file stl\_iterator\_base\_types.h.

## 4.781.3 Constructor &amp; Destructor Documentation

## 4.781.3.1 ostream\_iterator() [1/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 ostream_type & __s) [inline]
```

Construct from an ostream.

Definition at line 201 of file stream\_iterator.h.

## 4.781.3.2 ostream\_iterator() [2/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 ostream_type & __s,
 const _CharT * __c) [inline]
```

Construct from an ostream.

The delimiter string *c* is written to the stream after every *Tp* written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

## Parameters

|                                        |                                   |
|----------------------------------------|-----------------------------------|
| <a href="#"><code>_↵<br/>_s</code></a> | Underlying ostream to write to.   |
| <a href="#"><code>_↵<br/>_c</code></a> | CharT delimiter string to insert. |

Definition at line 214 of file `stream_iterator.h`.

4.781.3.3 `ostream_iterator()` [3/3]

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
 const ostream_iterator< _Tp, _CharT, _Traits > &__obj) [inline]
```

Copy constructor.

Definition at line 218 of file `stream_iterator.h`.

## 4.781.4 Member Function Documentation

4.781.4.1 `operator=()`

```
template<typename _Tp , typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator& std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
 const _Tp &__value) [inline]
```

Writes *value* to underlying ostream using `operator<<`. If constructed with delimiter string, writes delimiter to ostream.

Definition at line 228 of file `stream_iterator.h`.

The documentation for this class was generated from the following file:

- [stream\\_iterator.h](#)

4.782 `std::experimental::fundamentals_v2::ostream_joiner<_DelimT, _CharT, _Traits >` Class Template Reference

## Public Types

- typedef `_CharT` **char\_type**
- typedef void **difference\_type**
- typedef [output\\_iterator\\_tag](#) **iterator\_category**
- typedef [basic\\_ostream](#)< `_CharT`, `_Traits` > **ostream\_type**
- typedef void **pointer**
- typedef void **reference**
- typedef `_Traits` **traits\_type**
- typedef void **value\_type**

## Public Member Functions

- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, const \_DelimT &\_\_delimiter) noexcept(is\_nothrow\_copy\_constructible\_v<\_v<\_DelimT>)
- **ostream\_joiner** ([ostream\\_type](#) &\_\_os, \_DelimT &&\_\_delimiter) noexcept(is\_nothrow\_move\_constructible\_v<\_DelimT>)
- [ostream\\_joiner](#) & **operator\*** () noexcept
- [ostream\\_joiner](#) & **operator++** () noexcept
- [ostream\\_joiner](#) & **operator++** (int) noexcept
- template<typename \_Tp >  
[ostream\\_joiner](#) & **operator=** (const \_Tp &\_\_value)

## 4.782.1 Detailed Description

```
template<typename _DelimT, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >
```

Output iterator that inserts a delimiter between elements.

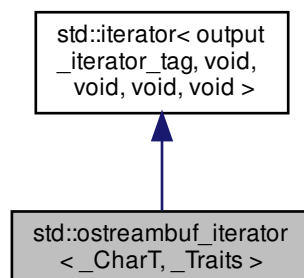
Definition at line 58 of file experimental/iterator.

The documentation for this class was generated from the following file:

- [experimental/iterator](#)

## 4.783 std::ostreambuf\_iterator&lt; \_CharT, \_Traits &gt; Class Template Reference

Inheritance diagram for std::ostreambuf\_iterator< \_CharT, \_Traits >:



### Public Types

- typedef void [difference\\_type](#)
  - typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
  - typedef void [pointer](#)
  - typedef void [reference](#)
  - typedef void [value\\_type](#)
- 
- typedef [\\_CharT](#) [char\\_type](#)
  - typedef [\\_Traits](#) [traits\\_type](#)
  - typedef [basic\\_streambuf](#)< [\\_CharT](#), [\\_Traits](#) > [streambuf\\_type](#)
  - typedef [basic\\_ostream](#)< [\\_CharT](#), [\\_Traits](#) > [ostream\\_type](#)

### Public Member Functions

- [ostreambuf\\_iterator](#) ([ostream\\_type](#) &\_\_s) noexcept
- [ostreambuf\\_iterator](#) ([streambuf\\_type](#) \* \_\_s) noexcept
- [ostreambuf\\_iterator](#) & [\\_M\\_put](#) (const [\\_CharT](#) \* \_\_ws, [streamsize](#) \_\_len)
- bool [failed](#) () const noexcept
- [ostreambuf\\_iterator](#) & [operator\\*](#) ()
- [ostreambuf\\_iterator](#) & [operator++](#) (int)
- [ostreambuf\\_iterator](#) & [operator++](#) ()
- [ostreambuf\\_iterator](#) & [operator=](#) ([\\_CharT](#) \_\_c)

### Friends

- template<typename [\\_CharT2](#) >  
    [\\_\\_gnu\\_cxx::\\_\\_enable\\_if](#)< [\\_\\_is\\_char](#)< [\\_CharT2](#) >::\_\_value, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >::\_\_type **copy**  
    ([istreambuf\\_iterator](#)< [\\_CharT2](#) >, [istreambuf\\_iterator](#)< [\\_CharT2](#) >, [ostreambuf\\_iterator](#)< [\\_CharT2](#) >)

#### 4.783.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::ostreambuf_iterator< _CharT, _Traits >
```

Provides output iterator semantics for streambufs.

Definition at line 128 of file iosfwd.

#### 4.783.2 Member Typedef Documentation

#### 4.783.2.1 char\_type

```
template<typename _CharT , typename _Traits >
typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

Definition at line 249 of file ostreambuf\_iterator.h.

#### 4.783.2.2 difference\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file stl\_iterator\_base\_types.h.

#### 4.783.2.3 iterator\_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file stl\_iterator\_base\_types.h.

#### 4.783.2.4 ostream\_type

```
template<typename _CharT , typename _Traits >
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
```

Public typedefs.

Definition at line 252 of file ostreambuf\_iterator.h.

#### 4.783.2.5 pointer

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file stl\_iterator\_base\_types.h.

#### 4.783.2.6 reference

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file `stl_iterator_base_types.h`.

#### 4.783.2.7 streambuf\_type

```
template<typename _CharT , typename _Traits >
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf_type
```

Public typedefs.

Definition at line 251 of file `streambuf_iterator.h`.

#### 4.783.2.8 traits\_type

```
template<typename _CharT , typename _Traits >
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
```

Public typedefs.

Definition at line 250 of file `streambuf_iterator.h`.

#### 4.783.2.9 value\_type

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

### 4.783.3 Constructor & Destructor Documentation



#### 4.783.3.1 ostreambuf\_iterator() [1/2]

```
template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
 ostream_type & __s) [inline], [noexcept]
```

Construct output iterator from ostream.

Definition at line 274 of file ostreambuf\_iterator.h.

#### 4.783.3.2 ostreambuf\_iterator() [2/2]

```
template<typename _CharT , typename _Traits >
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
 streambuf_type * __s) [inline], [noexcept]
```

Construct output iterator from streambuf.

Definition at line 278 of file ostreambuf\_iterator.h.

### 4.783.4 Member Function Documentation

#### 4.783.4.1 failed()

```
template<typename _CharT , typename _Traits >
bool std::ostreambuf_iterator< _CharT, _Traits >::failed () const [inline], [noexcept]
```

Return true if previous operator=() failed.

Definition at line 308 of file ostreambuf\_iterator.h.

#### 4.783.4.2 operator\*()

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator* () [inline]
```

Return \*this.

Definition at line 293 of file ostreambuf\_iterator.h.

**4.783.4.3 operator++()** [1/2]

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
 int) [inline]
```

Return \*this.

Definition at line 298 of file ostreambuf\_iterator.h.

**4.783.4.4 operator++()** [2/2]

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator++ () [inline]
```

Return \*this.

Definition at line 303 of file ostreambuf\_iterator.h.

**4.783.4.5 operator=()**

```
template<typename _CharT , typename _Traits >
ostreambuf_iterator& std::ostreambuf_iterator< _CharT, _Traits >::operator= (
 _CharT __c) [inline]
```

Write character to ostreambuf. Calls ostreambuf.sputc().

Definition at line 283 of file ostreambuf\_iterator.h.

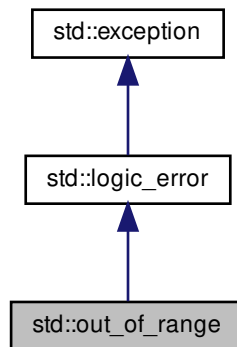
References `std::basic_ostreambuf< _CharT, _Traits >::sputc()`.

The documentation for this class was generated from the following files:

- [iosfwd](#)
- [ostreambuf\\_iterator.h](#)

## 4.784 std::out\_of\_range Class Reference

Inheritance diagram for std::out\_of\_range:



## Public Member Functions

- **out\_of\_range** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **out\_of\_range** (const char \*) \_GLIBCXX\_TXN\_SAFE
- **out\_of\_range** (const [out\\_of\\_range](#) &)=default
- **out\_of\_range** ([out\\_of\\_range](#) &&)=default
- [out\\_of\\_range](#) & **operator=** (const [out\\_of\\_range](#) &)=default
- [out\\_of\\_range](#) & **operator=** ([out\\_of\\_range](#) &&)=default
- virtual const char \* [what](#) () const noexcept

## 4.784.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in `basic_string`).

Definition at line 200 of file `stdexcept`.

## 4.784.2 Member Function Documentation

#### 4.784.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 4.785 `std::output_iterator_tag` Struct Reference

#### 4.785.1 Detailed Description

Marking output iterators.

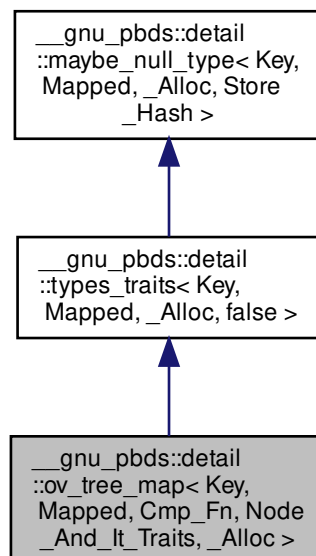
Definition at line 96 of file `stl_iterator_base_types.h`.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

### 4.786 `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`:



## Classes

- class `cond_dtor`

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `point_const_iterator` **const\_iterator**
- typedef `traits_base::const_pointer` **const\_pointer**
- typedef `traits_base::const_reference` **const\_reference**
- typedef `ov_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `point_iterator` **iterator**
- typedef `traits_base::key_const_pointer` **key\_const\_pointer**
- typedef `traits_base::key_const_reference` **key\_const\_reference**
- typedef `traits_base::key_pointer` **key\_pointer**
- typedef `traits_base::key_reference` **key\_reference**
- typedef `traits_base::key_type` **key\_type**
- typedef `traits_base::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `traits_base::mapped_const_reference` **mapped\_const\_reference**
- typedef `traits_base::mapped_pointer` **mapped\_pointer**
- typedef `traits_base::mapped_reference` **mapped\_reference**
- typedef `traits_base::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `traits_type::node_update` **node\_update**
- typedef `const_pointer` **point\_const\_iterator**
- typedef `pointer` **point\_iterator**
- typedef `traits_base::pointer` **pointer**
- typedef `traits_base::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `traits_base::value_type` **value\_type**

## Public Member Functions

- **ov\_tree\_map** (const `Cmp_Fn` &)
- **ov\_tree\_map** (const `Cmp_Fn` &, const `node_update` &)
- **ov\_tree\_map** (const `ov_tree_map`< `Key`, `Mapped`, `Cmp_Fn`, `Node_And_It_Traits`, `_Alloc` > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename `It` >  
void **copy\_from\_range** (`It`, `It`)
- bool **empty** () const

- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator it)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference r\_key)
- point\_const\_iterator **find** (key\_const\_reference r\_key) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- [std::pair](#)< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** ([ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference r\_key)
- point\_const\_iterator **lower\_bound** (key\_const\_reference r\_key) const
- size\_type **max\_size** () const
- node\_const\_iterator [node\\_begin](#) () const
- node\_iterator [node\\_begin](#) ()
- node\_const\_iterator [node\\_end](#) () const
- node\_iterator [node\\_end](#) ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- size\_type **size** () const
- void **split** (key\_const\_reference, [ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** ([ov\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference r\_key)
- point\_const\_iterator **upper\_bound** (key\_const\_reference r\_key) const

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### 4.786.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Ordered-vector tree associative-container.

Definition at line 106 of file `ov_tree_map.hpp`.

#### 4.786.2 Member Function Documentation

**4.786.2.1 node\_begin()** [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 47 of file ov\_tree\_map.hpp.

**4.786.2.2 node\_begin()** [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () [inline]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 59 of file ov\_tree\_map.hpp.

**4.786.2.3 node\_end()** [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 53 of file ov\_tree\_map.hpp.

**4.786.2.4 node\_end()** [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

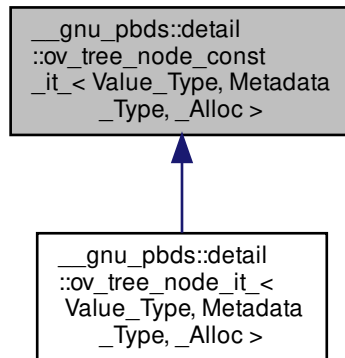
Definition at line 65 of file ov\_tree\_map.hpp.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map.hpp](#)

#### 4.787 `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >`:



##### Public Types

- typedef `rebind_traits< _Alloc, typename remove_const< Value_Type >::type >::const_pointer` **const\_reference**
- typedef `trivial_iterator_difference_type` **difference\_type**
- typedef `trivial_iterator_tag` **iterator\_category**
- typedef `rebind_traits< _Alloc, metadata_type >::const_reference` **metadata\_const\_reference**
- typedef `Metadata_Type` **metadata\_type**
- typedef `rebind_traits< _Alloc, typename remove_const< Value_Type >::type >::const_pointer` **reference**
- typedef `rebind_traits< _Alloc, Value_Type >::const_pointer` **value\_type**

##### Public Member Functions

- **ov\_tree\_node\_const\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_metadata\_pointer p\_metadata=0)
- `this_type get_l_child ()` const
- metadata\_const\_reference **get\_metadata** () const
- `this_type get_r_child ()` const
- bool **operator!=** (const `this_type` &other) const
- const\_reference **operator\*** () const
- bool **operator==** (const `this_type` &other) const

##### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**



## Protected Types

- typedef [rebind\\_traits](#)< `_Alloc`, `Metadata_Type` >::const\_pointer **const\_metadata\_pointer**
- typedef [rebind\\_traits](#)< `_Alloc`, `Value_Type` >::const\_pointer **const\_pointer**
- typedef [rebind\\_traits](#)< `_Alloc`, `Value_Type` >::pointer **pointer**
- typedef [ov\\_tree\\_node\\_const\\_it\\_](#)< `Value_Type`, `Metadata_Type`, `_Alloc` > **this\_type**

## Static Protected Member Functions

- template<typename `Ptr` >  
static `Ptr` **mid\_pointer** (`Ptr` p\_begin, `Ptr` p\_end)

### 4.787.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >
```

Const node reference.

Definition at line 57 of file `ov_tree_map_/node_iterators.hpp`.

### 4.787.2 Member Function Documentation

#### 4.787.2.1 `get_l_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↔
l_child () const [inline]
```

Returns the node iterator associated with the left node.

Definition at line 135 of file `ov_tree_map_/node_iterators.hpp`.

#### 4.787.2.2 `get_r_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↔
r_child () const [inline]
```

Returns the node iterator associated with the right node.

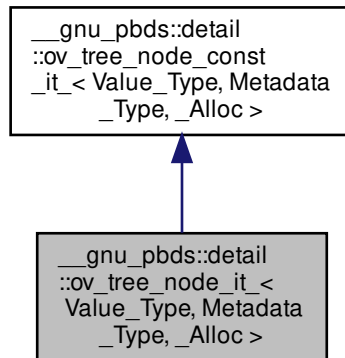
Definition at line 151 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

#### 4.788 `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >`:



##### Public Types

- typedef `rebind_traits< _Alloc, typename remove_const< Value_Type >::type >::pointer` **const\_reference**
- typedef `trivial_iterator_difference_type` **difference\_type**
- typedef `trivial_iterator_tag` **iterator\_category**
- typedef `rebind_traits< _Alloc, metadata_type >::const_reference` **metadata\_const\_reference**
- typedef `Metadata_Type` **metadata\_type**
- typedef `rebind_traits< _Alloc, typename remove_const< Value_Type >::type >::pointer` **reference**
- typedef `rebind_traits< _Alloc, Value_Type >::pointer` **value\_type**

##### Public Member Functions

- **ov\_tree\_node\_it\_** (const\_pointer p\_nd=0, const\_pointer p\_begin\_nd=0, const\_pointer p\_end\_nd=0, const\_↔ metadata\_pointer p\_metadata=0)
- `ov_tree_node_it_ get_l_child` () const
- metadata\_const\_reference **get\_metadata** () const
- `ov_tree_node_it_ get_r_child` () const
- bool **operator!=** (const `this_type` &other) const
- reference `operator*` () const
- bool **operator==** (const `this_type` &other) const

##### Public Attributes

- pointer **m\_p\_begin\_value**
- pointer **m\_p\_end\_value**
- const\_metadata\_pointer **m\_p\_metadata**
- pointer **m\_p\_value**

## Static Protected Member Functions

- `template<typename Ptr >`  
`static Ptr mid_pointer (Ptr p_begin, Ptr p_end)`

### 4.788.1 Detailed Description

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >
```

Node reference.

Definition at line 197 of file `ov_tree_map_/node_iterators.hpp`.

### 4.788.2 Member Function Documentation

#### 4.788.2.1 `get_l_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_l_child () const [inline]
```

Returns the node reference associated with the left node.

Definition at line 239 of file `ov_tree_map_/node_iterators.hpp`.

#### 4.788.2.2 `get_r_child()`

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↵
_r_child () const [inline]
```

Returns the node reference associated with the right node.

Definition at line 255 of file `ov_tree_map_/node_iterators.hpp`.

#### 4.788.2.3 operator\*()

```
template<typename Value_Type , typename Metadata_Type , typename _Alloc >
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* (
) const [inline]
```

Access.

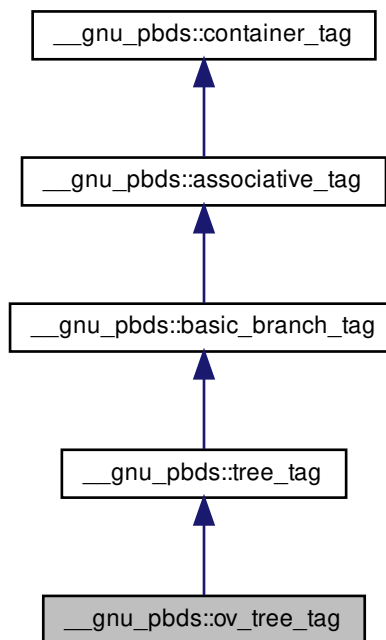
Definition at line 234 of file `ov_tree_map_/node_iterators.hpp`.

The documentation for this class was generated from the following file:

- [ov\\_tree\\_map\\_/node\\_iterators.hpp](#)

#### 4.789 \_\_gnu\_pbds::ov\_tree\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::ov_tree_tag`:



##### 4.789.1 Detailed Description

Ordered-vector tree.

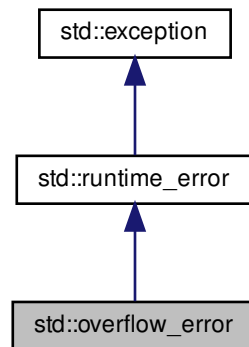
Definition at line 159 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.790 `std::overflow_error` Class Reference

Inheritance diagram for `std::overflow_error`:



### Public Member Functions

- **`overflow_error`** (const [string](#) &\_\_arg) `_GLIBCXX_TXN_SAFE`
- **`overflow_error`** (const char \*) `_GLIBCXX_TXN_SAFE`
- **`overflow_error`** (const [overflow\\_error](#) &)=default
- **`overflow_error`** ([overflow\\_error](#) &&)=default
- [overflow\\_error](#) & **`operator=`** (const [overflow\\_error](#) &)=default
- [overflow\\_error](#) & **`operator=`** ([overflow\\_error](#) &&)=default
- virtual const char \* [what](#) () const noexcept

### 4.790.1 Detailed Description

Thrown to indicate arithmetic overflow.

Definition at line 273 of file `stdexcept`.

### 4.790.2 Member Function Documentation

#### 4.790.2.1 `what()`

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

### 4.791 `std::owner_less<_Tp>` Struct Template Reference

#### 4.791.1 Detailed Description

```
template<typename _Tp = void>
struct std::owner_less<_Tp>
```

Primary template `owner_less`.

Definition at line 768 of file `bits/shared_ptr.h`.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

### 4.792 `std::experimental::fundamentals_v2::owner_less<shared_ptr<_Tp>>` Struct Template Reference

Inherits `std::_Sp_owner_less<_Tp, _Tp1>`.

#### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

#### Public Member Functions

- `bool` **operator()** (const `_Tp` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept
- `bool` **operator()** (const `_Tp` &\_\_lhs, const `_Tp1` &\_\_rhs) const noexcept
- `bool` **operator()** (const `_Tp1` &\_\_lhs, const `_Tp` &\_\_rhs) const noexcept

#### 4.792.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
```

Partial specialization of owner\_less for shared\_ptr.

Definition at line 509 of file experimental/bits/shared\_ptr.h.

#### 4.792.2 Member Typedef Documentation

##### 4.792.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 4.792.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

##### 4.792.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

#### 4.793 std::owner\_less< shared\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const noexcept`

#### 4.793.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >
```

Partial specialization of `owner_less` for `shared_ptr`.

Definition at line 777 of file `bits/shared_ptr.h`.

#### 4.793.2 Member Typedef Documentation

##### 4.793.2.1 `first_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.793.2.2 `result_type`

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.



## 4.793.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 4.794 std::owner\_less&lt; void &gt; Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

## Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

## Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

## 4.794.1 Detailed Description

```
template<>
struct std::owner_less< void >
```

Void specialization of owner\_less compares either shared\_ptr or weak\_ptr.

Definition at line 772 of file bits/shared\_ptr.h.

## 4.794.2 Member Typedef Documentation

#### 4.794.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.794.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.794.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

### 4.795 std::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

Inherits std::Sp\_owner\_less< \_Tp, \_Tp1 >.

#### Public Types

- typedef \_Tp [first\\_argument\\_type](#)
- typedef bool [result\\_type](#)
- typedef \_Tp [second\\_argument\\_type](#)

#### Public Member Functions

- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp &\_\_lhs, const \_Tp1 &\_\_rhs) const noexcept
- bool **operator()** (const \_Tp1 &\_\_lhs, const \_Tp &\_\_rhs) const noexcept

### 4.795.1 Detailed Description

```
template<typename _Tp>
struct std::owner_less< weak_ptr< _Tp > >
```

Partial specialization of owner\_less for weak\_ptr.

Definition at line 783 of file bits/shared\_ptr.h.

### 4.795.2 Member Typedef Documentation

#### 4.795.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.795.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.795.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 4.796 std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > > Struct Template Reference

Inherits std::\_Sp\_owner\_less< \_Tp, \_Tp1 >.

### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `bool` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- `bool` **operator()** (`const _Tp &__lhs, const _Tp &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp &__lhs, const _Tp1 &__rhs`) `const noexcept`
- `bool` **operator()** (`const _Tp1 &__lhs, const _Tp &__rhs`) `const noexcept`

#### 4.796.1 Detailed Description

```
template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >
```

Partial specialization of `owner_less` for `weak_ptr`.

Definition at line 515 of file `experimental/bits/shared_ptr.h`.

#### 4.796.2 Member Typedef Documentation

##### 4.796.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.796.2.2 result\_type

```
typedef bool std::binary_function< _Tp , _Tp , bool >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

4.796.2.3 `second_argument_type`

```
typedef _Tp std::binary_function< _Tp , _Tp , bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

Definition at line 124 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [experimental/bits/shared\\_ptr.h](#)

4.797 `std::packaged_task< _Res(_ArgTypes...)>` Class Template Reference

## Public Member Functions

- `template<typename _Fn, typename = __not_same<_Fn>>`  
**`packaged_task`** (`_Fn` && `_fn`)
- `template<typename _Fn, typename _Alloc, typename = __not_same<_Fn>>`  
**`packaged_task`** (`allocator_arg_t`, `const _Alloc` & `_a`, `_Fn` && `_fn`)
- `template<typename _Allocator >`  
**`packaged_task`** (`allocator_arg_t`, `const _Allocator` & `_a`) `noexcept`
- `template<typename _Allocator >`  
**`packaged_task`** (`allocator_arg_t`, `const _Allocator` &, `const packaged_task` &) `=delete`
- `template<typename _Allocator >`  
**`packaged_task`** (`allocator_arg_t`, `const _Allocator` &, `packaged_task` && `_other`) `noexcept`
- **`packaged_task`** (`const packaged_task` &) `=delete`
- **`packaged_task`** (`packaged_task` && `_other`) `noexcept`
- `future< _Res >` **`get_future`** ()
- `void` **`make_ready_at_thread_exit`** (`_ArgTypes...` `__args`)
- `void` **`operator()`** (`_ArgTypes...` `__args`)
- `packaged_task` & **`operator=`** (`const packaged_task` &) `=delete`
- `packaged_task` & **`operator=`** (`packaged_task` && `_other`) `noexcept`
- `void` **`reset`** ()
- `void` **`swap`** (`packaged_task` & `_other`) `noexcept`
- `bool` **`valid`** () `const noexcept`

## 4.797.1 Detailed Description

```
template<typename _Res, typename... _ArgTypes>
class std::packaged_task< _Res(_ArgTypes...)>
```

`packaged_task`

Definition at line 1481 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

## 4.798 `std::pair<_T1,_T2>` Struct Template Reference

Inherits `__pair_base<_T1,_T2>`.

### Public Types

- typedef `_T1` [first\\_type](#)
- typedef `_T2` [second\\_type](#)

### Public Member Functions

- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if<__and<__is_implicitly_default_constructible<_U1>, __is_implicitly_default_constructible<_U2>>::value, bool>::type = true>`  
`constexpr pair()`
- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if<_PCCP::template _ConstructiblePair<_U1,_U2>() &&_PCCP::template _ImplicitlyConvertiblePair<_U1,_U2>(), bool>::type = true>`  
`constexpr pair(const _T1 &__a, const _T2 &__b)`
- `template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if<_PCCP::template _ConstructiblePair<_U1,_U2>() &&!_PCCP::template _ImplicitlyConvertiblePair<_U1,_U2>(), bool>::type = false>`  
`constexpr pair(const _T1 &__a, const _T2 &__b)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1,_U2>::template _ConstructiblePair<_U1,_U2>() &&_PCCFP<_U1,_U2>::template _ImplicitlyConvertiblePair<_U1,_U2>(), bool>::type = true>`  
`constexpr pair(const pair<_U1,_U2> &__p)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1,_U2>::template _ConstructiblePair<_U1,_U2>() &&!_PCCFP<_U1,_U2>::template _ImplicitlyConvertiblePair<_U1,_U2>(), bool>::type = false>`  
`constexpr pair(const pair<_U1,_U2> &__p)`
- `constexpr pair(const pair &)=default`
- `constexpr pair(pair &&)=default`
- `template<typename _U1, typename enable_if<_PCCP::template _MoveCopyPair<true,_U1,_T2>(), bool>::type = true>`  
`constexpr pair(_U1 &&__x, const _T2 &__y)`
- `template<typename _U1, typename enable_if<_PCCP::template _MoveCopyPair<false,_U1,_T2>(), bool>::type = false>`  
`constexpr pair(_U1 &&__x, const _T2 &__y)`
- `template<typename _U2, typename enable_if<_PCCP::template _CopyMovePair<true,_T1,_U2>(), bool>::type = true>`  
`constexpr pair(const _T1 &__x, _U2 &&__y)`
- `template<typename _U2, typename enable_if<_PCCP::template _CopyMovePair<false,_T1,_U2>(), bool>::type = false>`  
`pair(const _T1 &__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCP::template _MoveConstructiblePair<_U1,_U2>() &&_PCCP::template _ImplicitlyMoveConvertiblePair<_U1,_U2>(), bool>::type = true>`  
`constexpr pair(_U1 &&__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCP::template _MoveConstructiblePair<_U1,_U2>() &&!_PCCP::template _ImplicitlyMoveConvertiblePair<_U1,_U2>(), bool>::type = false>`  
`constexpr pair(_U1 &&__x, _U2 &&__y)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1,_U2>::template _MoveConstructiblePair<_U1,_U2>() &&_PCCFP<_U1,_U2>::template _ImplicitlyMoveConvertiblePair<_U1,_U2>(), bool>::type = true>`  
`constexpr pair(pair<_U1,_U2> &&__p)`
- `template<typename _U1, typename _U2, typename enable_if<_PCCFP<_U1,_U2>::template _MoveConstructiblePair<_U1,_U2>() &&!_PCCFP<_U1,_U2>::template _ImplicitlyMoveConvertiblePair<_U1,_U2>(), bool>::type = false>`  
`constexpr pair(pair<_U1,_U2> &&__p)`
- `template<typename... _Args1, typename... _Args2>`  
`constexpr pair(piecewise_construct_t, tuple<_Args1...>, tuple<_Args2...>)`

- constexpr **pair** & **operator=** (typename conditional<\_\_and\_< is\_copy\_assignable<\_T1>, is\_copy\_assignable<\_T2>>::value, const **pair** &, const \_\_nonesuch & >::type \_\_p)
- constexpr **pair** & **operator=** (typename conditional<\_\_and\_< is\_move\_assignable<\_T1>, is\_move\_assignable<\_T2>>::value, **pair** &&, \_\_nonesuch && >::type \_\_p) noexcept(\_\_and\_< is\_nothrow\_move\_assignable<\_T1>, is\_nothrow\_move\_assignable<\_T2>>::value)
- template<typename \_U1, typename \_U2 >  
constexpr **enable\_if**<\_\_and\_< is\_assignable<\_T1 &, const \_U1 & >, is\_assignable<\_T2 &, const \_U2 & >>::value, **pair** & >::type **operator=** (const **pair**<\_U1, \_U2> &\_\_p)
- template<typename \_U1, typename \_U2 >  
constexpr **enable\_if**<\_\_and\_< is\_assignable<\_T1 &, \_U1 && >, is\_assignable<\_T2 &, \_U2 && >>::value, **pair** & >::type **operator=** (**pair**<\_U1, \_U2> &&\_\_p)
- constexpr void **swap** (**pair** &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable<\_T1>, \_\_is\_nothrow\_swappable<\_T2>>::value)

#### Public Attributes

- **\_T1** **first**
- **\_T2** **second**

#### Related Functions

(Note that these are not member functions.)

- template<typename \_T1, typename \_T2 >  
constexpr **pair**< typename \_\_decay\_and\_strip<\_T1>::\_\_type, typename \_\_decay\_and\_strip<\_T2>::\_\_type > **make\_pair** (\_T1 &&\_\_x, \_T2 &&\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator==** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator<** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator!=** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator>** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator<=** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr bool **operator>=** (const **pair**<\_T1, \_T2> &\_\_x, const **pair**<\_T1, \_T2> &\_\_y)
- template<typename \_T1, typename \_T2 >  
constexpr **enable\_if**<\_\_and\_< \_\_is\_swappable<\_T1>, \_\_is\_swappable<\_T2>>::value >::type **swap** (**pair**<\_T1, \_T2> &\_\_x, **pair**<\_T1, \_T2> &\_\_y) noexcept(noexcept(\_\_x.swap(\_\_y)))

#### 4.798.1 Detailed Description

```
template<typename _T1, typename _T2>
struct std::pair<_T1, _T2>
```

Struct holding two objects of arbitrary type.

### Template Parameters

|                  |                        |
|------------------|------------------------|
| <code>_T1</code> | Type of first object.  |
| <code>_T2</code> | Type of second object. |

<https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html>

Definition at line 211 of file `stl_pair.h`.

## 4.798.2 Member Typedef Documentation

### 4.798.2.1 `first_type`

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the `first` member.

Definition at line 214 of file `stl_pair.h`.

### 4.798.2.2 `second_type`

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the `second` member.

Definition at line 215 of file `stl_pair.h`.

## 4.798.3 Constructor & Destructor Documentation

### 4.798.3.1 `pair()` [1/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and< __is_implicitly_↵
default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 >> ::value, bool >↵
::type = true>
constexpr std::pair< _T1, _T2 >::pair () [inline]
```

The default constructor creates `first` and `second` using their respective default constructors.

Definition at line 232 of file `stl_pair.h`.



## 4.798.3.2 pair() [2/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
constexpr std::pair< _T1, _T2 >::pair (
 const _T1 & __a,
 const _T2 & __b) [inline]
```

Construct from two const lvalues, allowing implicit conversions.

Definition at line 266 of file stl\_pair.h.

## 4.798.3.3 pair() [3/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
constexpr std::pair< _T1, _T2 >::pair (
 const _T1 & __a,
 const _T2 & __b) [inline], [explicit]
```

Construct from two const lvalues, disallowing implicit conversions.

Definition at line 276 of file stl\_pair.h.

## 4.798.3.4 pair() [4/5]

```
template<typename _T1, typename _T2>
constexpr std::pair< _T1, _T2 >::pair (
 const pair< _T1, _T2 > &) [default]
```

Copy constructor.

## 4.798.3.5 pair() [5/5]

```
template<typename _T1, typename _T2>
constexpr std::pair< _T1, _T2 >::pair (
 pair< _T1, _T2 > &&) [default]
```

Move constructor.

#### 4.798.4 Member Function Documentation

##### 4.798.4.1 swap()

```
template<typename _T1, typename _T2>
constexpr void std::pair< _T1, _T2 >::swap (
 pair< _T1, _T2 > & __p) [inline], [noexcept]
```

Swap the first members and then the second members.

Definition at line 439 of file `stl_pair.h`.

Referenced by `std::pair< _Biliter, _Biliter >::swap()`.

#### 4.798.5 Member Data Documentation

##### 4.798.5.1 first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.

Definition at line 217 of file `stl_pair.h`.

Referenced by `__gnu_parallel::__find_template()`, `std::__sample()`, `std::Temporary_buffer< _ForwardIterator, _Tp >::Temporary_buffer()`, `std::set< _Key, _Compare, _Alloc >::insert()`, `std::pair< _Biliter, _Biliter >::operator<()`, `std::pair< _Biliter, _Biliter >::operator==()`, and `std::pair< _Biliter, _Biliter >::swap()`.

##### 4.798.5.2 second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

Definition at line 218 of file `stl_pair.h`.

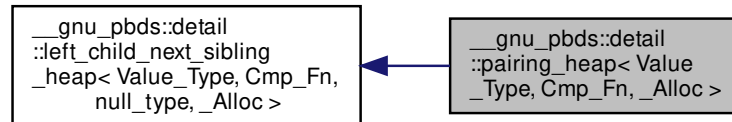
Referenced by `std::__sample()`, `std::set< _Key, _Compare, _Alloc >::insert()`, `std::pair< _Biliter, _Biliter >::operator<()`, `std::pair< _Biliter, _Biliter >::operator==()`, and `std::pair< _Biliter, _Biliter >::swap()`.

The documentation for this struct was generated from the following files:

- [stl\\_pair.h](#)
- [tuple](#)

4.799 `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc >` **node**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**

## Public Member Functions

- **pairing\_heap** (const `Cmp_Fn` &)
- **pairing\_heap** (const `pairing_heap` &)
- **iterator** **begin** ()
- `const_iterator` **begin** () const
- void **clear** ()
- bool **empty** () const
- `iterator` **end** ()
- `const_iterator` **end** () const
- void **erase** (`point_iterator`)
- template<typename `Pred` >  
size\_type **erase\_if** (`Pred`)
- `Cmp_Fn` & **get\_cmp\_fn** ()
- const `Cmp_Fn` & **get\_cmp\_fn** () const
- void **join** (`pairing_heap` &)

- size\_type **max\_size** () const
- void **modify** (point\_iterator, const\_reference)
- void **pop** ()
- point\_iterator **push** (const\_reference)
- size\_type **size** () const
- template<typename Pred >  
void **split** (Pred, pairing\_heap &)
- void **swap** (pairing\_heap &)
- void **swap** (left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, null\_type, \_Alloc > &)
- const\_reference **top** () const

### Protected Types

- typedef alloc\_traits::allocator\_type **node\_allocator**
- typedef alloc\_traits::const\_pointer **node\_const\_pointer**
- typedef null\_type **node\_metadata**
- typedef std::pair< node\_pointer, node\_pointer > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **bubble\_to\_top** (node\_pointer)
- void **clear\_imp** (node\_pointer)
- template<typename It >  
void **copy\_from\_range** (It, It)
- node\_pointer **get\_new\_node\_for\_insert** (const\_reference)
- node\_pointer **prune** (Pred)
- void **swap\_with\_parent** (node\_pointer, node\_pointer)
- void **to\_linked\_list** ()
- void **value\_swap** (left\_child\_next\_sibling\_heap &)

### Static Protected Member Functions

- static void **make\_child\_of** (node\_pointer, node\_pointer)
- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

## 4.799.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
```

Pairing heap.

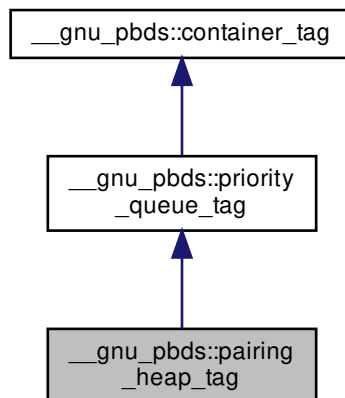
Definition at line 77 of file pairing\_heap.hpp.

The documentation for this class was generated from the following file:

- [pairing\\_heap.hpp](#)

## 4.800 \_\_gnu\_pbds::pairing\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pairing\_heap\_tag:



## 4.800.1 Detailed Description

Pairing-heap.

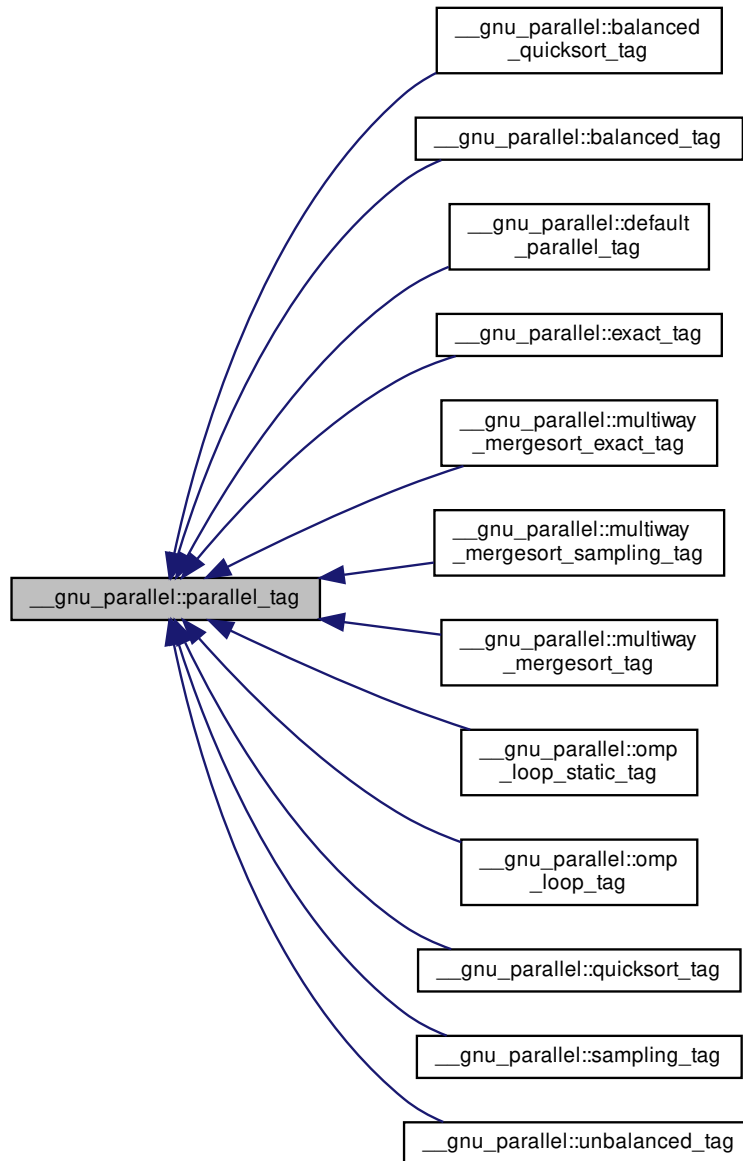
Definition at line 174 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.801 \_\_gnu\_parallel::parallel\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::parallel\_tag:



#### Public Member Functions

- `parallel_tag ()`
- `parallel_tag (_ThreadIndex __num_threads)`
- `_ThreadIndex __get_num_threads ()`
- `void set_num_threads (_ThreadIndex __num_threads)`

#### 4.801.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

Definition at line 46 of file tags.h.

#### 4.801.2 Constructor & Destructor Documentation

##### 4.801.2.1 parallel\_tag() [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag () [inline]
```

Default constructor. Use default number of threads.

Definition at line 53 of file tags.h.

##### 4.801.2.2 parallel\_tag() [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
 _ThreadIndex __num_threads) [inline]
```

Default constructor. Recommend number of threads to use.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 58 of file tags.h.

#### 4.801.3 Member Function Documentation

##### 4.801.3.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by \_\_gnu\_parallel::\_\_parallel\_sort().

#### 4.801.3.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline]
```

Set the desired number of threads.

##### Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.802 std::poisson\_distribution<\_IntType>::param\_type Struct Reference

##### Public Types

- typedef [poisson\\_distribution](#)<\_IntType> **distribution\_type**

##### Public Member Functions

- **param\_type** (double \_\_mean)
- double **mean** () const

##### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class **poisson\_distribution**<\_IntType>

##### 4.802.1 Detailed Description

```
template<typename _IntType = int>
struct std::poisson_distribution<_IntType>::param_type
```

Parameter type.

Definition at line 4429 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)



## 4.803 std::chi\_squared\_distribution<\_RealType>::param\_type Struct Reference

### Public Types

- typedef [chi\\_squared\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_n)
- \_RealType **n** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.803.1 Detailed Description

```
template<typename _RealType = double>
struct std::chi_squared_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2640 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.804 std::binomial\_distribution<\_IntType>::param\_type Struct Reference

### Public Types

- typedef [binomial\\_distribution](#)<\_IntType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_IntType \_\_t, double \_\_p=0.5)
- double **p** () const
- \_IntType **t** () const

## Friends

- class **binomial\_distribution**< \_IntType >
- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.804.1 Detailed Description

```
template<typename _IntType = int>
struct std::binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3748 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.805 std::lognormal\_distribution&lt; \_RealType &gt;::param\_type Struct Reference

## Public Types

- typedef [lognormal\\_distribution](#)< \_RealType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_RealType \_\_m, \_RealType \_\_s=\_RealType(1))
- \_RealType **m** () const
- \_RealType **s** () const

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.805.1 Detailed Description

```
template<typename _RealType = double>
struct std::lognormal_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 2201 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.806 std::student\_t\_distribution<\_RealType>::param\_type Struct Reference

### Public Types

- typedef [student\\_t\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_n)
- \_RealType **n** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.806.1 Detailed Description

```
template<typename _RealType = double>
struct std::student_t_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 3304 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.807 std::extreme\_value\_distribution<\_RealType>::param\_type Struct Reference

### Public Types

- typedef [extreme\\_value\\_distribution](#)<\_RealType> **distribution\_type**

### Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1.0))
- \_RealType **a** () const
- \_RealType **b** () const

### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.807.1 Detailed Description

```
template<typename _RealType = double>
struct std::extreme_value_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 5080 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

### 4.808 std::discrete\_distribution< \_IntType >::param\_type Struct Reference

#### Public Types

- typedef [discrete\\_distribution](#)< \_IntType > **distribution\_type**

#### Public Member Functions

- template<typename \_InputIterator >  
**param\_type** (\_InputIterator \_\_wbegin, \_InputIterator \_\_wend)
- **param\_type** ([initializer\\_list](#)< double > \_\_wil)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, double \_\_xmin, double \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default
- [std::vector](#)< double > **probabilities** () const

#### Friends

- class **discrete\_distribution**< \_IntType >
- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.808.1 Detailed Description

```
template<typename _IntType = int>
struct std::discrete_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 5287 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.809 std::piecewise\_linear\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef [piecewise\\_linear\\_distribution](#)<\_RealType> **distribution\_type**

## Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**param\_type** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**param\_type** ([initializer\\_list](#)<\_RealType> \_\_bl, \_Func \_\_fw)
- template<typename \_Func >  
**param\_type** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **param\_type** (const [param\\_type](#) &)=default
- [std::vector](#)<double> **densities** () const
- [std::vector](#)<\_RealType> **intervals** () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class [piecewise\\_linear\\_distribution](#)<\_RealType>

## 4.809.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_linear_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 5793 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.810 std::fisher\_f\_distribution&lt;\_RealType&gt;::param\_type Struct Reference

## Public Types

- typedef [fisher\\_f\\_distribution](#)<\_RealType> **distribution\_type**

#### Public Member Functions

- **param\_type** (\_RealType \_\_m, \_RealType \_\_n=\_RealType(1))
- \_RealType **m** () const
- \_RealType **n** () const

#### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.810.1 Detailed Description

```
template<typename _RealType = double>
struct std::fisher_f_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 3072 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.811 std::bernoulli\_distribution::param\_type Struct Reference

##### Public Types

- typedef [bernoulli\\_distribution](#) **distribution\_type**

##### Public Member Functions

- **param\_type** (double \_\_p)
- double **p** () const

##### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.811.1 Detailed Description

Parameter type.

Definition at line 3528 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.812 `std::exponential_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `exponential_distribution<_RealType>` **distribution\_type**

## Public Member Functions

- **param\_type** (`_RealType __lambda`)
- `_RealType lambda` () const

## Friends

- bool **operator!=** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

## 4.812.1 Detailed Description

```
template<typename _RealType = double>
struct std::exponential_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 4655 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

4.813 `std::weibull_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef `weibull_distribution<_RealType>` **distribution\_type**

#### Public Member Functions

- **param\_type** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1.0))
- \_RealType **a** () const
- \_RealType **b** () const

#### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

#### 4.813.1 Detailed Description

```
template<typename _RealType = double>
struct std::weibull_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 4870 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.814 std::normal\_distribution< \_RealType >::param\_type Struct Reference

##### Public Types

- typedef [normal\\_distribution](#)< \_RealType > **distribution\_type**

##### Public Member Functions

- **param\_type** (\_RealType \_\_mean, \_RealType \_\_stddev=\_RealType(1))
- \_RealType **mean** () const
- \_RealType **stddev** () const

#### Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)



## 4.814.1 Detailed Description

```
template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type
```

Parameter type.

Definition at line 1980 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

## 4.815 std::negative\_binomial\_distribution&lt; \_IntType &gt;::param\_type Struct Reference

## Public Types

- typedef [negative\\_binomial\\_distribution](#)< \_IntType > **distribution\_type**

## Public Member Functions

- **param\_type** (\_IntType \_\_k, double \_\_p=0.5)
- \_IntType **k** () const
- double **p** () const

## Friends

- bool **operator!=** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool **operator==** (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.815.1 Detailed Description

```
template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 4198 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.816 `std::geometric_distribution< _IntType >::param_type` Struct Reference

##### Public Types

- typedef `geometric_distribution< _IntType >` **distribution\_type**

##### Public Member Functions

- **param\_type** (double \_\_p)
- double **p** () const

##### Friends

- class **geometric\_distribution< \_IntType >**
- bool **operator!=** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool **operator==** (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

##### 4.816.1 Detailed Description

```
template<typename _IntType = int>
struct std::geometric_distribution< _IntType >::param_type
```

Parameter type.

Definition at line 3988 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.817 `std::gamma_distribution< _RealType >::param_type` Struct Reference

##### Public Types

- typedef `gamma_distribution< _RealType >` **distribution\_type**

##### Public Member Functions

- **param\_type** (\_RealType \_\_alpha\_val, \_RealType \_\_beta\_val=\_RealType(1))
- \_RealType **alpha** () const
- \_RealType **beta** () const

## Friends

- class `gamma_distribution<_RealType>`
- bool `operator!=` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool `operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.817.1 Detailed Description

```
template<typename _RealType = double>
struct std::gamma_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2412 of file `random.h`.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.818 `std::cauchy_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [cauchy\\_distribution<\\_RealType>](#) `distribution_type`

## Public Member Functions

- `param_type` (`_RealType` \_\_a, `_RealType` \_\_b=`_RealType`(1))
- `_RealType a` () const
- `_RealType b` () const

## Friends

- bool `operator!=` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- bool `operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.818.1 Detailed Description

```
template<typename _RealType = double>
struct std::cauchy_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 2864 of file `random.h`.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.819 `std::uniform_real_distribution<_RealType>::param_type` Struct Reference

##### Public Types

- typedef `uniform_real_distribution<_RealType>` `distribution_type`

##### Public Member Functions

- `param_type` (`_RealType` \_\_a, `_RealType` \_\_b=`_RealType`(1))
- `result_type a` () const
- `result_type b` () const

##### Friends

- bool `operator!=` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)
- bool `operator==` (const `param_type` &\_\_p1, const `param_type` &\_\_p2)

##### 4.819.1 Detailed Description

```
template<typename _RealType = double>
struct std::uniform_real_distribution<_RealType>::param_type
```

Parameter type.

Definition at line 1750 of file random.h.

The documentation for this struct was generated from the following file:

- [random.h](#)

#### 4.820 `std::uniform_int_distribution<_IntType>::param_type` Struct Reference

##### Public Types

- typedef `uniform_int_distribution<_IntType>` `distribution_type`

##### Public Member Functions

- `param_type` (`_IntType` \_\_a, `_IntType` \_\_b=`numeric_limits<_IntType>::max()`)
- `result_type a` () const
- `result_type b` () const

## Friends

- `bool operator!=` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- `bool operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)

## 4.820.1 Detailed Description

```
template<typename _IntType = int>
struct std::uniform_int_distribution<_IntType>::param_type
```

Parameter type.

Definition at line 83 of file `uniform_int_dist.h`.

The documentation for this struct was generated from the following file:

- [uniform\\_int\\_dist.h](#)

4.821 `std::piecewise_constant_distribution<_RealType>::param_type` Struct Reference

## Public Types

- typedef [piecewise\\_constant\\_distribution<\\_RealType>](#) **distribution\_type**

## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW>`  
**param\_type** (`_InputIteratorB __bfirst`, `_InputIteratorB __bend`, `_InputIteratorW __wbegin`)
- `template<typename _Func>`  
**param\_type** ([initializer\\_list](#)<\_RealType> \_\_bi, `_Func __fw`)
- `template<typename _Func>`  
**param\_type** (`size_t __nw`, `_RealType __xmin`, `_RealType __xmax`, `_Func __fw`)
- **param\_type** (const [param\\_type](#) &)=default
- `std::vector< double > densities` () const
- `std::vector< _RealType > intervals` () const
- [param\\_type](#) & **operator=** (const [param\\_type](#) &)=default

## Friends

- `bool operator!=` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- `bool operator==` (const [param\\_type](#) &\_\_p1, const [param\\_type](#) &\_\_p2)
- class `piecewise_constant_distribution<_RealType>`

#### 4.821.1 Detailed Description

```
template<typename _RealType = double>
struct std::piecewise_constant_distribution< _RealType >::param_type
```

Parameter type.

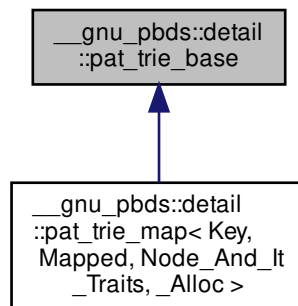
Definition at line 5522 of file random.h.

The documentation for this struct was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

#### 4.822 \_\_gnu\_pbds::detail::pat\_trie\_base Struct Reference

Inheritance diagram for \_\_gnu\_pbds::detail::pat\_trie\_base:



#### Classes

- class [\\_CIter](#)
- struct [\\_Head](#)
- struct [\\_Inode](#)
- class [\\_Iter](#)
- struct [\\_Leaf](#)
- struct [\\_Metadata](#)
- struct [\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_Node\\_base](#)
- class [\\_Node\\_citer](#)
- class [\\_Node\\_iter](#)

## Public Types

- enum `node_type` { `i_node`, `leaf_node`, `head_node` }

### 4.822.1 Detailed Description

Base type for PATRICIA trees.

Definition at line 51 of file `pat_trie_base.hpp`.

### 4.822.2 Member Enumeration Documentation

#### 4.822.2.1 `node_type`

```
enum __gnu_pbds::detail::pat_trie_base::node_type
```

Three types of nodes.

`i_node` is used by `_Inode`, `leaf_node` by `_Leaf`, and `head_node` by `_Head`.

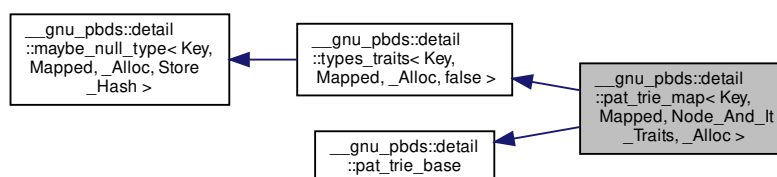
Definition at line 58 of file `pat_trie_base.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_base.hpp](#)

## 4.823 `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >`:



## Public Types

- typedef traits\_type::access\_traits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef std::pair< size\_type, size\_type > **comp\_hash**
- typedef point\_const\_iterator **const\_iterator**
- typedef traits\_base::const\_pointer **const\_pointer**
- typedef traits\_base::const\_reference **const\_reference**
- typedef traits\_type::const\_reverse\_iterator **const\_reverse\_iterator**
- typedef pat\_trie\_tag container\_category
- typedef \_Alloc::difference\_type **difference\_type**
- typedef point\_iterator **iterator**
- typedef traits\_base::key\_const\_pointer **key\_const\_pointer**
- typedef traits\_base::key\_const\_reference **key\_const\_reference**
- typedef traits\_base::key\_pointer **key\_pointer**
- typedef traits\_base::key\_reference **key\_reference**
- typedef traits\_base::key\_type **key\_type**
- typedef traits\_base::mapped\_const\_pointer **mapped\_const\_pointer**
- typedef traits\_base::mapped\_const\_reference **mapped\_const\_reference**
- typedef traits\_base::mapped\_pointer **mapped\_pointer**
- typedef traits\_base::mapped\_reference **mapped\_reference**
- typedef traits\_base::mapped\_type **mapped\_type**
- typedef \_\_nothrowcopy::indicator **no\_throw\_indicator**
- typedef traits\_type::node\_const\_iterator **node\_const\_iterator**
- typedef traits\_type::node\_iterator **node\_iterator**
- enum node\_type { i\_node, leaf\_node, head\_node }
- typedef traits\_type::node\_update **node\_update**
- typedef traits\_type::const\_iterator **point\_const\_iterator**
- typedef traits\_type::iterator **point\_iterator**
- typedef traits\_base::pointer **pointer**
- typedef traits\_base::reference **reference**
- typedef traits\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef stored\_data< value\_type, size\_type, Store\_Hash > **stored\_data\_type**
- typedef traits\_base::value\_type **value\_type**

## Public Member Functions

- **pat\_trie\_map** (const access\_traits &)
- **pat\_trie\_map** (const pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- const\_iterator **erase** (const\_iterator)
- iterator **erase** (iterator)



- `const_reverse_iterator` **erase** (`const_reverse_iterator`)
- `reverse_iterator` **erase** (`reverse_iterator`)
- `template<typename Pred >`  
`size_type` **erase\_if** (`Pred`)
- `point_iterator` **find** (`key_const_reference`)
- `point_const_iterator` **find** (`key_const_reference`) `const`
- `access_traits` & **get\_access\_traits** ()
- `const access_traits` & **get\_access\_traits** () `const`
- `node_update` & **get\_node\_update** ()
- `const node_update` & **get\_node\_update** () `const`
- `std::pair< point_iterator, bool >` **insert** (`const_reference`)
- `void` **join** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `point_iterator` **lower\_bound** (`key_const_reference`)
- `point_const_iterator` **lower\_bound** (`key_const_reference`) `const`
- `size_type` **max\_size** () `const`
- `node_const_iterator` **node\_begin** () `const`
- `node_iterator` **node\_begin** ()
- `node_const_iterator` **node\_end** () `const`
- `node_iterator` **node\_end** ()
- `mapped_reference` **operator[]** (`key_const_reference r_key`)
- `reverse_iterator` **rbegin** ()
- `const_reverse_iterator` **rbegin** () `const`
- `reverse_iterator` **rend** ()
- `const_reverse_iterator` **rend** () `const`
- `size_type` **size** () `const`
- `void` **split** (`key_const_reference`, `pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `void` **swap** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)
- `point_iterator` **upper\_bound** (`key_const_reference`)
- `point_const_iterator` **upper\_bound** (`key_const_reference`) `const`

#### Public Attributes

- `no_throw_indicator` **m\_no\_throw\_copies\_indicator**
- `store_extra` **m\_store\_extra\_indicator**

#### Protected Member Functions

- `template<typename It >`  
`void` **copy\_from\_range** (`It`, `It`)
- `node_pointer` **recursive\_copy\_node** (`node_const_pointer`)
- `void` **value\_swap** (`pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &`)

#### 4.823.1 Detailed Description

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >
```

PATRICIA trie.

This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000.

Definition at line 101 of file `pat_trie_.hpp`.

#### 4.823.2 Member Enumeration Documentation

##### 4.823.2.1 node\_type

```
enum __gnu_pbds::detail::pat_trie_base::node_type [inherited]
```

Three types of nodes.

i\_node is used by \_Inode, leaf\_node by \_Leaf, and head\_node by \_Head.

Definition at line 58 of file pat\_trie\_base.hpp.

#### 4.823.3 Member Function Documentation

##### 4.823.3.1 node\_begin() [1/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const [inline]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 103 of file pat\_trie\_.hpp.

##### 4.823.3.2 node\_begin() [2/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () [inline]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 109 of file pat\_trie\_.hpp.

##### 4.823.3.3 node\_end() [1/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const [inline]
```

Returns a const node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 115 of file pat\_trie\_.hpp.

## 4.823.3.4 node\_end() [2/2]

```
template<typename Key , typename Mapped , typename Node_And_It_Traits , typename _Alloc >
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () [inline]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

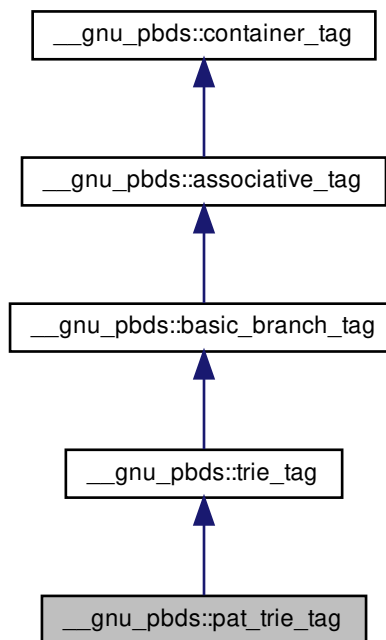
Definition at line 121 of file pat\_trie\_.hpp.

The documentation for this class was generated from the following file:

- [pat\\_trie\\_.hpp](#)

## 4.824 \_\_gnu\_pbds::pat\_trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::pat\_trie\_tag:



## 4.824.1 Detailed Description

PATRICIA trie.

Definition at line 165 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.825 std::experimental::filesystem::v1::path Class Reference

### Classes

- class [iterator](#)

### Public Types

- typedef [iterator](#) **const\_iterator**
- typedef [std::basic\\_string](#)< value\_type > **string\_type**
- typedef char **value\_type**

### Public Member Functions

- **path** (const [path](#) &\_\_p)=default
- **path** ([path](#) &&\_\_p) noexcept
- **path** ([string\\_type](#) &&\_\_source)
- template<typename \_Source , typename \_Require = \_\_detail::\_Path<\_Source>>  
**path** (\_Source const &\_\_source)
- template<typename \_InputIterator , typename \_Require = \_\_detail::\_Path<\_InputIterator, \_InputIterator>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- template<typename \_Source , typename \_Require = \_\_detail::\_Path<\_Source>, typename \_Require2 = \_\_detail::\_value\_type\_is\_↔  
char<\_Source>>  
**path** (\_Source const &\_\_source, const [locale](#) &\_\_loc)
- template<typename \_InputIterator , typename \_Require = \_\_detail::\_Path<\_InputIterator, \_InputIterator>, typename \_Require2 = \_\_↔  
detail::\_value\_type\_is\_char<\_InputIterator>>  
**path** (\_InputIterator \_\_first, \_InputIterator \_\_last, const [locale](#) &\_\_loc)
- template<typename \_Source >  
\_\_detail::\_Path<\_Source> & **append** (\_Source const &\_\_source)
- template<typename \_InputIterator >  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **append** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **path** & **assign** ([string\\_type](#) &&\_\_source)
- template<typename \_Source >  
\_\_detail::\_Path<\_Source> & **assign** (\_Source const &\_\_source)
- template<typename \_InputIterator >  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- **iterator begin** () const
- const value\_type \* **c\_str** () const noexcept
- void **clear** () noexcept
- int **compare** (const [path](#) &\_\_p) const noexcept
- int **compare** (const [string\\_type](#) &\_\_s) const
- int **compare** (const value\_type \*\_\_s) const
- int **compare** (const basic\_string\_view< value\_type > \_\_s) const
- template<typename \_Source >  
\_\_detail::\_Path<\_Source> & **concat** (\_Source const &\_\_x)
- template<typename \_InputIterator >  
\_\_detail::\_Path<\_InputIterator, \_InputIterator> & **concat** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- bool **empty** () const noexcept
- **iterator end** () const
- **path extension** () const

- [path](#) [filename](#) () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>> [std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **generic\_string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::string](#) **generic\_string** () const
- [std::u16string](#) **generic\_u16string** () const
- [std::u32string](#) **generic\_u32string** () const
- [std::string](#) **generic\_u8string** () const
- [std::wstring](#) **generic\_wstring** () const
- bool **has\_extension** () const
- bool **has\_filename** () const
- bool **has\_parent\_path** () const
- bool **has\_relative\_path** () const
- bool **has\_root\_directory** () const
- bool **has\_root\_name** () const
- bool **has\_root\_path** () const
- bool **has\_stem** () const
- bool **is\_absolute** () const
- bool **is\_relative** () const
- [path](#) & **make\_preferred** ()
- const [string\\_type](#) & **native** () const noexcept
- **operator string\_type** () const
- [path](#) & **operator+=** (const [path](#) &\_\_x)
- [path](#) & **operator+=** (const [string\\_type](#) &\_\_x)
- [path](#) & **operator+=** (const value\_type \*\_\_x)
- [path](#) & **operator+=** (value\_type \_\_x)
- [path](#) & **operator+=** (basic\_string\_view< value\_type > \_\_x)
- template<typename \_Source> [\\_\\_detail::Path](#)<\_Source> & **operator+=** (\_Source const &\_\_x)
- template<typename \_CharT> [\\_\\_detail::Path](#)<\_CharT\*, \_CharT\*> & **operator+=** (\_CharT \_\_x)
- [path](#) & **operator/=** (const [path](#) &\_\_p)
- template<typename \_Source> [\\_\\_detail::Path](#)<\_Source> & **operator/=** (\_Source const &\_\_source)
- [path](#) & **operator=** (const [path](#) &\_\_p)=default
- [path](#) & **operator=** ([path](#) &&\_\_p) noexcept
- [path](#) & **operator=** ([string\\_type](#) &&\_\_source)
- template<typename \_Source> [\\_\\_detail::Path](#)<\_Source> & **operator=** (\_Source const &\_\_source)
- [path](#) **parent\_path** () const
- [path](#) **relative\_path** () const
- [path](#) & **remove\_filename** ()
- [path](#) & **replace\_extension** (const [path](#) &\_\_replacement=[path](#)())
- [path](#) & **replace\_filename** (const [path](#) &\_\_replacement)
- [path](#) **root\_directory** () const
- [path](#) **root\_name** () const
- [path](#) **root\_path** () const
- [path](#) **stem** () const
- template<typename \_CharT, typename \_Traits = std::char\_traits<\_CharT>, typename \_Allocator = std::allocator<\_CharT>>> [std::basic\\_string](#)<\_CharT, \_Traits, \_Allocator> **string** (const \_Allocator &\_\_a=\_Allocator()) const
- [std::string](#) **string** () const
- void **swap** ([path](#) &\_\_rhs) noexcept

- [std::u16string](#) **u16string** () const
- [std::u32string](#) **u32string** () const
- [std::string](#) **u8string** () const
- [std::wstring](#) **wstring** () const

#### Static Public Attributes

- static constexpr value\_type **preferred\_separator**

#### 4.825.1 Detailed Description

A filesystem path.

Definition at line 195 of file `experimental/bits/fs_path.h`.

The documentation for this class was generated from the following file:

- [experimental/bits/fs\\_path.h](#)

### 4.826 `std::piecewise_constant_distribution<_RealType>` Class Template Reference

#### Classes

- struct [param\\_type](#)

#### Public Types

- typedef \_RealType [result\\_type](#)

#### Public Member Functions

- template<typename \_InputIteratorB, typename \_InputIteratorW >  
**piecewise\_constant\_distribution** (\_InputIteratorB \_\_bfirst, \_InputIteratorB \_\_bend, \_InputIteratorW \_\_wbegin)
- template<typename \_Func >  
**piecewise\_constant\_distribution** ([initializer\\_list](#)< \_RealType > \_\_bl, \_Func \_\_fw)
- template<typename \_Func >  
**piecewise\_constant\_distribution** (size\_t \_\_nw, \_RealType \_\_xmin, \_RealType \_\_xmax, \_Func \_\_fw)
- **piecewise\_constant\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)

- `std::vector< double > densities () const`
- `std::vector< _RealType > intervals () const`
- `result_type max () const`
- `result_type min () const`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng)`
- `template<typename _UniformRandomNumberGenerator >  
result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)`
- `param_type param () const`
- `void param (const param_type &__param)`
- `void reset ()`

#### Friends

- `template<typename _RealType1 , typename _CharT , typename _Traits >  
std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const  
std::piecewise_constant_distribution< _RealType1 > &__x)`
- `bool operator== (const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)`
- `template<typename _RealType1 , typename _CharT , typename _Traits >  
std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::piecewise_constant_distribution< _RealType1 > &__x)`

#### 4.826.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_constant_distribution< _RealType >
```

A piecewise\_constant\_distribution random number distribution.

The formula for the piecewise constant probability mass function is

Definition at line 5512 of file random.h.

#### 4.826.2 Member Typedef Documentation

##### 4.826.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 5515 of file random.h.

### 4.826.3 Member Function Documentation

#### 4.826.3.1 densities()

```
template<typename _RealType = double>
std::vector<double> std::piecewise_constant_distribution< _RealType >::densities () const [inline]
```

Returns a vector of the probability densities.

Definition at line 5637 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

#### 4.826.3.2 intervals()

```
template<typename _RealType = double>
std::vector<_RealType> std::piecewise_constant_distribution< _RealType >::intervals () const
[inline]
```

Returns a vector of the intervals.

Definition at line 5621 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`.

#### 4.826.3.3 max()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5672 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`, and `std::vector< _Tp, _Alloc >::empty()`.

#### 4.826.3.4 min()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5662 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::front()`.



## 4.826.3.5 operator()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::piecewise_constant_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 5683 of file random.h.

## 4.826.3.6 param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_constant_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5647 of file random.h.

## 4.826.3.7 param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5655 of file random.h.

## 4.826.3.8 reset()

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 5614 of file random.h.

## 4.826.4 Friends And Related Function Documentation

### 4.826.4.1 operator<<

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::piecewise_constant_distribution< _RealType1 > & __x) [friend]
```

Inserts a `piecewise_constant_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                          |
| <code>__x</code>  | A <code>piecewise_constant_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

### 4.826.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
 const piecewise_constant_distribution< _RealType > & __d1,
 const piecewise_constant_distribution< _RealType > & __d2) [friend]
```

Return true if two `piecewise constant distributions` have the same parameters.



Definition at line 5718 of file `random.h`.

### 4.826.4.3 operator>>

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::piecewise_constant_distribution< _RealType1 > & __x) [friend]
```

Extracts a `piecewise_constant_distribution` random number distribution `__x` from the input stream `__is`.

## Parameters

|                                                                                                      |                                                                   |
|------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| <a href="#">_is</a> | An input stream.                                                  |
| <a href="#">_x</a>  | A piecewise_constant_distribution random number generator engine. |

## Returns

The input stream with [!\[\]\(a03a7eb2f4046e1d3c76772003e549ea\_img.jpg\)\\_x](#) extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.827 std::piecewise\_construct\_t Struct Reference

## 4.827.1 Detailed Description

Tag type for piecewise construction of std::pair objects.

Definition at line 80 of file stl\_pair.h.

The documentation for this struct was generated from the following file:

- [stl\\_pair.h](#)

## 4.828 std::piecewise\_linear\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef [\\_RealType](#) [result\\_type](#)

## Public Member Functions

- `template<typename _InputIteratorB, typename _InputIteratorW >`  
**piecewise\_linear\_distribution** (`_InputIteratorB __bfirst`, `_InputIteratorB __bend`, `_InputIteratorW __wbegin`)
- `template<typename _Func >`  
**piecewise\_linear\_distribution** (`initializer_list<_RealType> __bl`, `_Func __fw`)
- `template<typename _Func >`  
**piecewise\_linear\_distribution** (`size_t __nw`, `_RealType __xmin`, `_RealType __xmax`, `_Func __fw`)
- **piecewise\_linear\_distribution** (`const param_type &__p`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
**void \_\_generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`)
- `template<typename _ForwardIterator, typename _UniformRandomNumberGenerator >`  
**void \_\_generate** (`_ForwardIterator __f`, `_ForwardIterator __t`, `_UniformRandomNumberGenerator &__urng`, `const param_type &__p`)
- `template<typename _UniformRandomNumberGenerator >`  
**void \_\_generate** (`result_type *__f`, `result_type *__t`, `_UniformRandomNumberGenerator &__urng`, `const param_type &__p`)
- `std::vector<double> densities` () const
- `std::vector<_RealType> intervals` () const
- `result_type max` () const
- `result_type min` () const
- `template<typename _UniformRandomNumberGenerator >`  
**result\_type operator()** (`_UniformRandomNumberGenerator &__urng`)
- `template<typename _UniformRandomNumberGenerator >`  
**result\_type operator()** (`_UniformRandomNumberGenerator &__urng`, `const param_type &__p`)
- `param_type param` () const
- `void param` (`const param_type &__param`)
- `void reset` ()

## Friends

- `template<typename _RealType1, typename _CharT, typename _Traits >`  
**std::basic\_ostream<\_CharT, \_Traits> & operator<<** (`std::basic_ostream<_CharT, _Traits> &__os`, `const std::piecewise_linear_distribution<_RealType1> &__x`)
- `bool operator==` (`const piecewise_linear_distribution &__d1`, `const piecewise_linear_distribution &__d2`)
- `template<typename _RealType1, typename _CharT, typename _Traits >`  
**std::basic\_istream<\_CharT, \_Traits> & operator>>** (`std::basic_istream<_CharT, _Traits> &__is`, `std::piecewise_linear_distribution<_RealType1> &__x`)

## 4.828.1 Detailed Description

```
template<typename _RealType = double>
class std::piecewise_linear_distribution<_RealType>
```

A `piecewise_linear_distribution` random number distribution.

The formula for the piecewise linear probability mass function is

Definition at line 5783 of file `random.h`.

## 4.828.2 Member Typedef Documentation

### 4.828.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 5786 of file random.h.

## 4.828.3 Member Function Documentation

### 4.828.3.1 densities()

```
template<typename _RealType = double>
std::vector<double> std::piecewise_linear_distribution< _RealType >::densities () const [inline]
```

Return a vector of the probability densities of the distribution.

Definition at line 5910 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

### 4.828.3.2 intervals()

```
template<typename _RealType = double>
std::vector<_RealType> std::piecewise_linear_distribution< _RealType >::intervals () const
[inline]
```

Return the intervals of the distribution.

Definition at line 5893 of file random.h.

References `std::vector<_Tp, _Alloc>::empty()`.

#### 4.828.3.3 max()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 5945 of file random.h.

References `std::vector< _Tp, _Alloc >::back()`, and `std::vector< _Tp, _Alloc >::empty()`.

#### 4.828.3.4 min()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 5935 of file random.h.

References `std::vector< _Tp, _Alloc >::empty()`, and `std::vector< _Tp, _Alloc >::front()`.

#### 4.828.3.5 operator>()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::piecewise_linear_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 5956 of file random.h.

#### 4.828.3.6 param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_linear_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 5920 of file random.h.

#### 4.828.3.7 param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 5928 of file random.h.

## 4.828.3.8 reset()

```
template<typename _RealType = double>
void std::piecewise_linear_distribution<_RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 5886 of file random.h.

## 4.828.4 Friends And Related Function Documentation

## 4.828.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream<_CharT, _Traits > & __os,
 const std::piecewise_linear_distribution<_RealType1 > & __x) [friend]
```

Inserts a piecewise\_linear\_distribution random number distribution `__x` into the output stream `__os`.

## Parameters

|                   |                                                             |
|-------------------|-------------------------------------------------------------|
| <code>__os</code> | An output stream.                                           |
| <code>__x</code>  | A piecewise_linear_distribution random number distribution. |

## Returns

The output stream with the state of `__x` inserted or in an error state.

## 4.828.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
```

```
const piecewise_linear_distribution< _RealType > & __d1,
const piecewise_linear_distribution< _RealType > & __d2) [friend]
```

Return true if two piecewise linear distributions have the same parameters.

Definition at line 5991 of file random.h.

#### 4.828.4.3 `operator>>`

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::piecewise_linear_distribution< _RealType1 > & __x) [friend]
```

Extracts a `piecewise_linear_distribution` random number distribution `__x` from the input stream `__is`.

##### Parameters

|                   |                                                                              |
|-------------------|------------------------------------------------------------------------------|
| <code>__is</code> | An input stream.                                                             |
| <code>__x</code>  | A <code>piecewise_linear_distribution</code> random number generator engine. |

##### Returns

The input stream with `__x` extracted or in an error state.

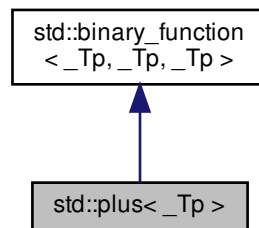
The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)



## 4.829 std::plus< \_Tp > Struct Template Reference

Inheritance diagram for std::plus< \_Tp >:



### Public Types

- typedef `_Tp` [first\\_argument\\_type](#)
- typedef `_Tp` [result\\_type](#)
- typedef `_Tp` [second\\_argument\\_type](#)

### Public Member Functions

- constexpr `_Tp` **operator()** (const `_Tp` &\_\_x, const `_Tp` &\_\_y) const

#### 4.829.1 Detailed Description

```
template<typename _Tp>
struct std::plus< _Tp >
```

One of the [math functors](#).

Definition at line 147 of file `stl_function.h`.

#### 4.829.2 Member Typedef Documentation

#### 4.829.2.1 first\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

#### 4.829.2.2 result\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

#### 4.829.2.3 second\_argument\_type

```
typedef _Tp std::binary_function< _Tp , _Tp , _Tp >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

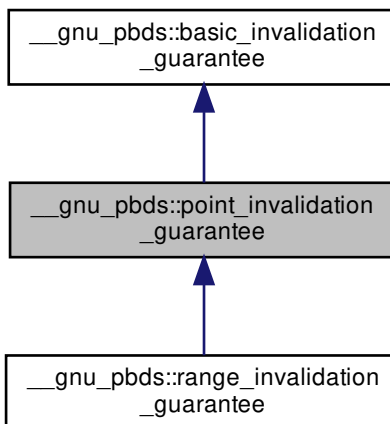
Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

### 4.830 \_\_gnu\_pbds::point\_invalidation\_guarantee Struct Reference

Inheritance diagram for \_\_gnu\_pbds::point\_invalidation\_guarantee:



## 4.830.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.

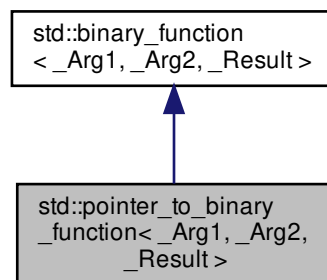
Definition at line 103 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.831 `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >` Class Template Reference

Inheritance diagram for `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`:



## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Result` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- **`pointer_to_binary_function`** (`_Result`(\*\_\_x)(`_Arg1`, `_Arg2`))
- `_Result` **`operator()`** (`_Arg1` \_\_x, `_Arg2` \_\_y) const

## Protected Attributes

- `_Result`(\* **`M_ptr`**)(`_Arg1`, `_Arg2`)

#### 4.831.1 Detailed Description

```
template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 1105 of file `stl_function.h`.

#### 4.831.2 Member Typedef Documentation

##### 4.831.2.1 first\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.831.2.2 result\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

##### 4.831.2.3 second\_argument\_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

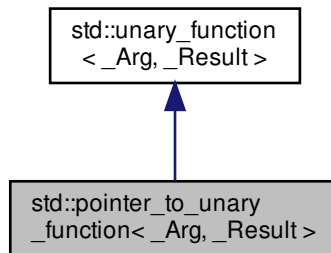
Definition at line 124 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

## 4.832 std::pointer\_to\_unary\_function< \_Arg, \_Result > Class Template Reference

Inheritance diagram for std::pointer\_to\_unary\_function< \_Arg, \_Result >:



### Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### Public Member Functions

- **pointer\_to\_unary\_function** (`_Result(*__x)(_Arg)`)
- `_Result` **operator()** (`_Arg __x`) const

### Protected Attributes

- `_Result(* M_ptr )(_Arg)`

#### 4.832.1 Detailed Description

```
template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >
```

One of the [adaptors for function pointers](#).

Definition at line 1080 of file `stl_function.h`.

#### 4.832.2 Member Typedef Documentation

#### 4.832.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.832.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

### 4.833 `std::pointer_traits<_Ptr>` Struct Template Reference

#### Public Types

- using [difference\\_type](#) = `__detected_or_t< ptrdiff_t, __difference_type, _Ptr >`
- using [element\\_type](#) = `__detected_or_t< __get_first_arg_t< _Ptr >, __element_type, _Ptr >`
- using [pointer](#) = `_Ptr`
- `template<typename _Up >`  
using [rebind](#) = `typename __rebind< _Ptr, _Up >::type`

#### Static Public Member Functions

- `static _Ptr pointer\_to (__make_not_void< element\_type > &__e)`

#### 4.833.1 Detailed Description

```
template<typename _Ptr>
struct std::pointer_traits< _Ptr >
```

Uniform interface to all pointer-like types.

Definition at line 83 of file `ptr_traits.h`.

#### 4.833.2 Member Typedef Documentation

##### 4.833.2.1 difference\_type

```
template<typename _Ptr>
using std::pointer_traits<_Ptr>::difference_type = __detected_or_t<ptrdiff_t, __difference_↵
type, _Ptr>
```

The type used to represent the difference between two pointers.

Definition at line 109 of file ptr\_traits.h.

##### 4.833.2.2 element\_type

```
template<typename _Ptr>
using std::pointer_traits<_Ptr>::element_type = __detected_or_t<__get_first_arg_t<_Ptr>, __↵
element_type, _Ptr>
```

The type pointed to.

Definition at line 105 of file ptr\_traits.h.

##### 4.833.2.3 pointer

```
template<typename _Ptr>
using std::pointer_traits<_Ptr>::pointer = _Ptr
```

The pointer type.

Definition at line 101 of file ptr\_traits.h.

##### 4.833.2.4 rebind

```
template<typename _Ptr>
template<typename _Up>
using std::pointer_traits<_Ptr>::rebind = typename __rebind<_Ptr, _Up>::type
```

A pointer to a different type.

Definition at line 113 of file ptr\_traits.h.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

## 4.834 `std::pointer_traits<_Tp*>` Struct Template Reference

### Public Types

- typedef ptrdiff\_t [difference\\_type](#)
- typedef \_Tp [element\\_type](#)
- typedef \_Tp\* [pointer](#)
- template<typename \_Up >  
using **rebind** = \_Up\*

### Static Public Member Functions

- static constexpr [pointer](#) [pointer\\_to](#) (\_\_make\_not\_void< [element\\_type](#) > &\_\_r) noexcept

#### 4.834.1 Detailed Description

```
template<typename _Tp>
struct std::pointer_traits<_Tp*>
```

Partial specialization for built-in pointers.

Definition at line 128 of file ptr\_traits.h.

#### 4.834.2 Member Typedef Documentation

##### 4.834.2.1 [difference\\_type](#)

```
template<typename _Tp >
typedef ptrdiff_t std::pointer_traits<_Tp*>::difference_type
```

Type used to represent the difference between two pointers.

Definition at line 135 of file ptr\_traits.h.

##### 4.834.2.2 [element\\_type](#)

```
template<typename _Tp >
typedef _Tp std::pointer_traits<_Tp*>::element_type
```

The type pointed to.

Definition at line 133 of file ptr\_traits.h.



4.834.2.3 `pointer`

```
template<typename _Tp >
typedef _Tp* std::pointer_traits< _Tp * >::pointer
```

The pointer type.

Definition at line 131 of file `ptr_traits.h`.

## 4.834.3 Member Function Documentation

4.834.3.1 `pointer_to()`

```
template<typename _Tp >
static constexpr pointer std::pointer_traits< _Tp * >::pointer_to (
 __make_not_void< element_type > & __r) [inline], [static], [noexcept]
```

Obtain a pointer to an object.

## Parameters

|                |                                                            |
|----------------|------------------------------------------------------------|
| <code>↔</code> | A reference to an object of type <code>element_type</code> |
| <code>↔</code> |                                                            |
| <code>↔</code> |                                                            |
| <code>↔</code> |                                                            |
| <code>r</code> |                                                            |

## Returns

```
addressof(__r)
```

Definition at line 146 of file `ptr_traits.h`.

References `std::addressof()`.

The documentation for this struct was generated from the following file:

- [ptr\\_traits.h](#)

4.835 `std::poisson_distribution<_IntType>` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` `result_type`

## Public Member Functions

- **poisson\_distribution** (double `__mean`)
- **poisson\_distribution** (const `param_type` &`__p`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_ForwardIterator` , typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`_ForwardIterator` `__f`, `_ForwardIterator` `__t`, `_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- template<typename `_UniformRandomNumberGenerator` >  
void **\_\_generate** (`result_type` \*`__f`, `result_type` \*`__t`, `_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- `result_type` **max** () const
- double **mean** () const
- `result_type` **min** () const
- template<typename `_UniformRandomNumberGenerator` >  
`result_type` **operator()** (`_UniformRandomNumberGenerator` &`__urng`)
- template<typename `_UniformRandomNumberGenerator` >  
`result_type` **operator()** (`_UniformRandomNumberGenerator` &`__urng`, const `param_type` &`__p`)
- `param_type` **param** () const
- void **param** (const `param_type` &`__param`)
- void **reset** ()

## Friends

- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >  
`std::basic_ostream`< `_CharT`, `_Traits` > & **operator<<** (`std::basic_ostream`< `_CharT`, `_Traits` > &`__os`, const `std::poisson_distribution`< `_IntType1` > &`__x`)
- bool **operator==** (const `poisson_distribution` &`__d1`, const `poisson_distribution` &`__d2`)
- template<typename `_IntType1` , typename `_CharT` , typename `_Traits` >  
`std::basic_istream`< `_CharT`, `_Traits` > & **operator>>** (`std::basic_istream`< `_CharT`, `_Traits` > &`__is`, `std::poisson_distribution`< `_IntType1` > &`__x`)

## 4.835.1 Detailed Description

```
template<typename _IntType = int>
class std::poisson_distribution< _IntType >
```

A discrete Poisson random number distribution.

The formula for the Poisson probability density function is  $p(i|\mu) = \frac{\mu^i}{i!} e^{-\mu}$  where  $\mu$  is the parameter of the distribution.

Definition at line 4419 of file random.h.

#### 4.835.2 Member Typedef Documentation

##### 4.835.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 4422 of file random.h.

#### 4.835.3 Member Function Documentation

##### 4.835.3.1 max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4523 of file random.h.

References std::numeric\_limits<\_Tp>::max().

##### 4.835.3.2 mean()

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean () const [inline]
```

Returns the distribution parameter mean.

Definition at line 4494 of file random.h.

##### 4.835.3.3 min()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4516 of file random.h.

**4.835.3.4 operator() [1/2]**

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::poisson_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 4531 of file random.h.

**4.835.3.5 operator() [2/2]**

```
template<typename _IntType >
template<typename _UniformRandomNumberGenerator >
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
 _UniformRandomNumberGenerator & __urng,
 const param_type & __param)
```

A rejection algorithm when mean  $\geq 12$  and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if `_GLIBCXX_USE_C99_MATH_TR1` is defined.

Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).

Definition at line 1269 of file bits/random.tcc.

References `std::abs()`, `std::numeric_limits<_Tp>::epsilon()`, `std::log()`, and `std::numeric_limits<_Tp>::max()`.

**4.835.3.6 param() [1/2]**

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4501 of file random.h.

**4.835.3.7 param() [2/2]**

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

#### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4509 of file random.h.

#### 4.835.3.8 reset()

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Definition at line 4487 of file random.h.

References `std::normal_distribution< _RealType >::reset()`.

#### 4.835.4 Friends And Related Function Documentation

##### 4.835.4.1 operator<<

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::poisson_distribution< _IntType1 > & __x) [friend]
```

Inserts a `poisson_distribution` random number distribution `__x` into the output stream `__os`.

#### Parameters

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| <code>__os</code> | An output stream.                                               |
| <code>__x</code>  | A <code>poisson_distribution</code> random number distribution. |

#### Returns

The output stream with the state of `__x` inserted or in an error state.

#### 4.835.4.2 operator==

```
template<typename _IntType = int>
bool operator== (
 const poisson_distribution< _IntType > & __d1,
 const poisson_distribution< _IntType > & __d2) [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.



Definition at line 4567 of file random.h.

#### 4.835.4.3 operator>>

```
template<typename _IntType = int>
template<typename _IntType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::poisson_distribution< _IntType1 > & __x) [friend]
```

Extracts a poisson\_distribution random number distribution \_\_x from the input stream \_\_is.

##### Parameters

|                                                                                             |                                                        |
|---------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <br>__is | An input stream.                                       |
| <br>__x  | A poisson_distribution random number generator engine. |

##### Returns

The input stream with \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.836 \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc > Class Template Reference

Inherits type< \_Tv, Cmp\_Fn, \_Alloc, Tag >.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `Tag` **container\_category**
- typedef `allocator_type::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `allocator_type::size_type` **size\_type**
- typedef `_Tv` **value\_type**

## Public Member Functions

- [priority\\_queue](#) (const `cmp_fn` &`r_cmp_fn`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`)
- `template<typename It >`  
[priority\\_queue](#) (It `first_it`, It `last_it`, const `cmp_fn` &`r_cmp_fn`)
- **priority\_queue** (const [priority\\_queue](#) &`other`)
- [priority\\_queue](#) & **operator=** (const [priority\\_queue](#) &`other`)
- void **swap** ([priority\\_queue](#) &`other`)

## 4.836.1 Detailed Description

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>
class __gnu_pbds::priority_queue<_Tv, Cmp_Fn, Tag, _Alloc>
```

A priority queue composed of one specific heap policy.

## Template Parameters

|                     |                                                                     |
|---------------------|---------------------------------------------------------------------|
| <code>_Tv</code>    | Value type.                                                         |
| <code>Cmp_Fn</code> | Comparison functor.                                                 |
| <code>Tag</code>    | Instantiating data structure type, see <code>container_tag</code> . |
| <code>_Alloc</code> | Allocator type.                                                     |

Base is dispatched at compile time via `Tag`, from the following choices: `binary_heap_tag`, `binomial_heap_tag`, `pairing_heap_tag`, `rc_binomial_heap_tag`, `thin_heap_tag`

Base choices are: `detail::binary_heap`, `detail::binomial_heap`, `detail::pairing_heap`, `detail::rc_binomial_heap`, `detail::thin_heap`.

Definition at line 83 of file priority\_queue.hpp.

The documentation for this class was generated from the following file:

- [priority\\_queue.hpp](#)

#### 4.837 std::priority\_queue< \_Tp, \_Sequence, \_Compare > Class Template Reference

##### Public Types

- typedef \_Sequence::const\_reference **const\_reference**
- typedef \_Sequence **container\_type**
- typedef \_Sequence::reference **reference**
- typedef \_Sequence::size\_type **size\_type**
- typedef \_Compare **value\_compare**
- typedef \_Sequence::value\_type **value\_type**

##### Public Member Functions

- template<typename \_Seq = \_Sequence, typename \_Requires = typename enable\_if<\_\_and<\_is\_default\_constructible<\_Compare>, is\_↵\_default\_constructible<\_Seq>>::value>::type>  
[priority\\_queue](#) ()
- **priority\_queue** (const \_Compare &\_\_x, const \_Sequence &\_\_s)
- **priority\_queue** (const \_Compare &\_\_x, \_Sequence &&\_\_s=\_Sequence())
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** (const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** (const \_Compare &\_\_x, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** (const \_Compare &\_\_x, const \_Sequence &\_\_c, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** (const \_Compare &\_\_x, \_Sequence &&\_\_c, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** (const [priority\\_queue](#) &\_\_q, const \_Alloc &\_\_a)
- template<typename \_Alloc, typename \_Requires = \_Uses<\_Alloc>>  
**priority\_queue** ([priority\\_queue](#) &&\_\_q, const \_Alloc &\_\_a)
- template<typename \_InputIterator >  
[priority\\_queue](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_x, const \_Sequence &\_\_s)
- template<typename \_InputIterator >  
**priority\_queue** (\_InputIterator \_\_first, \_InputIterator \_\_last, const \_Compare &\_\_x=\_Compare(), \_Sequence &&\_\_s=\_Sequence())
- template<typename... \_Args>  
void **emplace** (\_Args &&... \_\_args)
- bool [empty](#) () const
- void [pop](#) ()
- void [push](#) (const value\_type &\_\_x)
- void **push** (value\_type &&\_\_x)
- size\_type [size](#) () const
- void **swap** ([priority\\_queue](#) &\_\_pq) noexcept(\_\_and<\_is\_nothrow\_swappable<\_↵Sequence>, \_is\_nothrow\_swappable<\_Compare>>::value)
- const\_reference [top](#) () const



### Protected Attributes

- `_Sequence` **c**
- `_Compare` **comp**

#### 4.837.1 Detailed Description

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue<_Tp, _Sequence, _Compare>
```

A standard container automatically sorting its contents.

### Template Parameters

|                        |                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                                              |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>vector&lt;_Tp&gt;</code> .                     |
| <code>_Compare</code>  | Comparison function object type, defaults to <code>less&lt;_Sequence::value_type&gt;</code> . |

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::vector`, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as `std::deque` or an appropriate user-defined type.

The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard queue operations.

### Note

No equality/comparison operators are provided for `priority_queue`.

Sorting of the elements takes place as they are added to, and removed from, the `priority_queue` using the `priority_queue`'s member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the `priority_queue` will not re-sort the elements for you. (How could it know to do so?)

Definition at line 456 of file `stl_queue.h`.

#### 4.837.2 Constructor & Destructor Documentation

**4.837.2.1** `priority_queue()` [1/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and<is_default_↵
constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue () [inline]
```

Default constructor creates no elements.

Definition at line 514 of file `stl_queue.h`.

**4.837.2.2** `priority_queue()` [2/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
template<typename _InputIterator >
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __x,
 const _Sequence & __s) [inline]
```

Builds a queue from a range.

**Parameters**

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| <code>__first</code> | An input iterator.                                      |
| <code>__last</code>  | An input iterator.                                      |
| <code>__x</code>     | A comparison functor describing a strict weak ordering. |
| <code>__s</code>     | An initial sequence with which to start.                |

Begins by copying `__s`, inserting a copy of the elements from `[first,last)` into the copy of `__s`, then ordering the copy according to `__x`.

For more information on function objects, see the documentation on [functor base classes](#).

Definition at line 586 of file `stl_queue.h`.

**4.837.3** Member Function Documentation

## 4.837.3.1 empty()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
bool std::priority_queue<_Tp, _Sequence, _Compare >::empty () const [inline]
```

Returns true if the queue is empty.

Definition at line 612 of file stl\_queue.h.

## 4.837.3.2 pop()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
void std::priority_queue<_Tp, _Sequence, _Compare >::pop () [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 675 of file stl\_queue.h.

## 4.837.3.3 push()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↵
Sequence::value_type>>
void std::priority_queue<_Tp, _Sequence, _Compare >::push (
 const value_type & __x) [inline]
```

Add data to the queue.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>_↵</code>  | Data to be added. |
| <code>__x</code> |                   |

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.

Definition at line 640 of file stl\_queue.h.

References std::push\_heap().

#### 4.837.3.4 size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
size_type std::priority_queue< _Tp, _Sequence, _Compare >::size () const [inline]
```

Returns the number of elements in the queue.

Definition at line 617 of file `stl_queue.h`.

#### 4.837.3.5 top()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top () const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

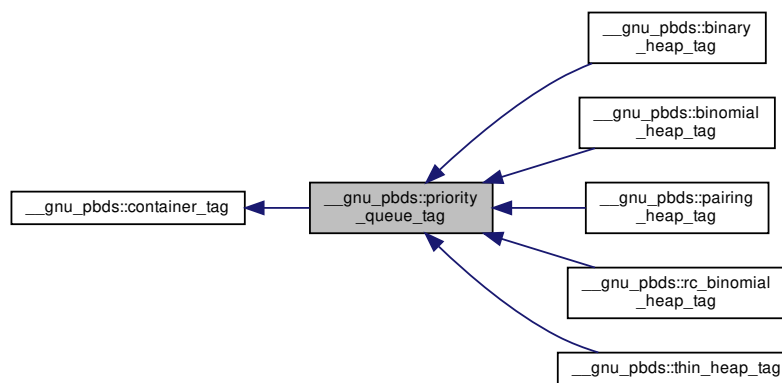
Definition at line 625 of file `stl_queue.h`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

#### 4.838 \_\_gnu\_pbds::priority\_queue\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::priority_queue_tag`:



## 4.838.1 Detailed Description

Basic priority-queue.

Definition at line 171 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.839 `__gnu_pbds::detail::probe_fn_base<_Alloc>` Class Template Reference

## 4.839.1 Detailed Description

```
template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base<_Alloc>
```

Probe functor base.

Definition at line 52 of file `probe_fn_base.hpp`.

The documentation for this class was generated from the following file:

- [probe\\_fn\\_base.hpp](#)

4.840 `__gnu_cxx::project1st<_Arg1, _Arg2>` Struct Template Reference

Inherits `__gnu_cxx::_Project1st<_Arg1, _Arg2>`.

## Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Arg1` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

## Public Member Functions

- `_Arg1` **operator()** (const `_Arg1` &`__x`, const `_Arg2` &) const

#### 4.840.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 229 of file ext/functional.

#### 4.840.2 Member Typedef Documentation

##### 4.840.2.1 first\_argument\_type

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::first_argument_type [inherited]
```

first\_argument\_type is the type of the first argument

Definition at line 121 of file stl\_function.h.

##### 4.840.2.2 result\_type

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::result_type [inherited]
```

result\_type is the return type

Definition at line 127 of file stl\_function.h.

##### 4.840.2.3 second\_argument\_type

```
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg1 >::second_argument_type [inherited]
```

second\_argument\_type is the type of the second argument

Definition at line 124 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

#### 4.841 \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 > Struct Template Reference

Inherits `__gnu_cxx::Project2nd< _Arg1, _Arg2 >`.

### Public Types

- typedef `_Arg1` [first\\_argument\\_type](#)
- typedef `_Arg2` [result\\_type](#)
- typedef `_Arg2` [second\\_argument\\_type](#)

### Public Member Functions

- `_Arg2` **operator()** (const `_Arg1` &, const `_Arg2` &\_\_y) const

#### 4.841.1 Detailed Description

```
template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
```

An [SGI extension](#) .

Definition at line 233 of file `ext/functional`.

#### 4.841.2 Member Typedef Documentation

##### 4.841.2.1 `first_argument_type`

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg2 >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

Definition at line 121 of file `stl_function.h`.

##### 4.841.2.2 `result_type`

```
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg2 >::result_type [inherited]
```

`result_type` is the return type

Definition at line 127 of file `stl_function.h`.

#### 4.841.2.3 second\_argument\_type

typedef [\\_Arg2](#) [std::binary\\_function](#)< [\\_Arg1](#), [\\_Arg2](#), [\\_Arg2](#) >::[second\\_argument\\_type](#) [inherited]

[second\\_argument\\_type](#) is the type of the second argument

Definition at line 124 of file [stl\\_function.h](#).

The documentation for this struct was generated from the following file:

- [ext/functional](#)

### 4.842 std::promise<\_Res> Class Template Reference

#### Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- [template](#)<typename [\\_Allocator](#) >  
**promise** ([allocator\\_arg\\_t](#), const [\\_Allocator](#) &\_\_a)
- [template](#)<typename [\\_Allocator](#) >  
**promise** ([allocator\\_arg\\_t](#), const [\\_Allocator](#) &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)< [\\_Res](#) > **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** (const [\\_Res](#) &\_\_r)
- void **set\_value** ([\\_Res](#) &&\_\_r)
- void **set\_value\_at\_thread\_exit** (const [\\_Res](#) &\_\_r)
- void **set\_value\_at\_thread\_exit** ([\\_Res](#) &&\_\_r)
- void **swap** ([promise](#) &\_\_rhs) noexcept

#### Friends

- [template](#)<typename , typename >  
class [\\_State](#)::**\_Setter**

#### 4.842.1 Detailed Description

[template](#)<typename [\\_Res](#)>

class [std::promise](#)< [\\_Res](#) >

Primary template for [promise](#).

Definition at line 134 of file [future](#).

The documentation for this class was generated from the following file:

- [future](#)



4.843 `std::promise<_Res &>` Class Template Reference

## Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &\_\_a)
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)<\_Res &> **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** (\_Res &\_\_r)
- void **set\_value\_at\_thread\_exit** (\_Res &\_\_r)
- void **swap** ([promise](#) &\_\_rhs) noexcept

## Friends

- `template<typename , typename >`  
class **\_State::Setter**

## 4.843.1 Detailed Description

`template<typename _Res>`  
class `std::promise<_Res &>`

Partial specialization for `promise<R&>`

Definition at line 1165 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

4.844 `std::promise< void >` Class Template Reference

## Public Member Functions

- **promise** ([promise](#) &&\_\_rhs) noexcept
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &\_\_a)
- `template<typename _Allocator >`  
**promise** ([allocator\\_arg\\_t](#), const \_Allocator &, [promise](#) &&\_\_rhs)
- **promise** (const [promise](#) &)=delete
- [future](#)< void > **get\_future** ()
- [promise](#) & **operator=** ([promise](#) &&\_\_rhs) noexcept
- [promise](#) & **operator=** (const [promise](#) &)=delete
- void **set\_exception** ([exception\\_ptr](#) \_\_p)
- void **set\_exception\_at\_thread\_exit** ([exception\\_ptr](#) \_\_p)
- void **set\_value** ()
- void **set\_value\_at\_thread\_exit** ()
- void **swap** ([promise](#) &\_\_rhs) noexcept

## Friends

- `template<typename , typename >`  
`class _State::_Setter`

## 4.844.1 Detailed Description

`template<>`  
`class std::promise< void >`

Explicit specialization for `promise<void>`

Definition at line 1255 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

4.845 `std::experimental::fundamentals_v2::propagate_const<_Tp>` Class Template Reference

## Public Types

- `typedef remove\_reference\_t< decltype(*std::declval<_Tp &>)> element_type`

## Public Member Functions

- `propagate_const (const propagate\_const &__p)=delete`
- `constexpr propagate_const (propagate\_const &&__p)=default`
- `template<typename _Up , typename enable_if< __and< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >>::value, bool >::type = true>`  
`constexpr propagate_const (propagate\_const< _Up > &&__pu)`
- `template<typename _Up , typename enable_if< __and< is_constructible< _Tp, _Up && >, __not< is_convertible< _Up &&, _Tp >>::value, bool >::type = false>`  
`constexpr propagate_const (propagate\_const< _Up > &&__pu)`
- `template<typename _Up , typename enable_if< __and< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >, __not< __is_propagate_const< typename decay< _Up >::type >>::value, bool >::type = true>`  
`constexpr propagate_const (_Up &&__u)`
- `template<typename _Up , typename enable_if< __and< is_constructible< _Tp, _Up && >, __not< is_convertible< _Up &&, _Tp >>, __not< __is_propagate_const< typename decay< _Up >::type >>::value, bool >::type = false>`  
`constexpr propagate_const (_Up &&__u)`
- `constexpr const element_type * get () const`
- `constexpr element_type * get ()`
- `constexpr operator bool () const`
- `template<typename _Up = _Tp, typename enable_if< __or< is_pointer< _Up >, is_convertible< _Up, const element_type * >>::value, bool >::type = true>`  
`constexpr operator const element_type * () const`

- `template<typename _Up = _Tp, typename enable_if< __or_< is_pointer< _Up >, is_convertible< _Up, const element_type * > >::value, bool >::type = true>`  
`constexpr operator element_type * ()`
- `constexpr const element_type & operator* () const`
- `constexpr element_type & operator* ()`
- `constexpr const element_type * operator-> () const`
- `constexpr element_type * operator-> ()`
- `propagate\_const & operator= (const propagate\_const &__p)=delete`
- `constexpr propagate\_const & operator= (propagate\_const &&__p)=default`
- `template<typename _Up, typename = typename enable_if<is_convertible<_Up&&, _Tp>::value>::type>`  
`constexpr propagate\_const & operator= (propagate\_const < _Up > &&__pu)`
- `template<typename _Up, typename = typename enable_if<__and_<is_convertible<_Up&&, _Tp>, __not_<__is_propagate_const<`  
`typename decay<_Up>::type>> >::value>::type>`  
`constexpr propagate\_const & operator= (_Up &&__u)`
- `constexpr void swap (propagate\_const &__pt) noexcept(__is_nothrow_swappable< _Tp >::value)`

#### Friends

- `template<typename _Up >`  
`constexpr const _Up & get_underlying (const propagate\_const < _Up > &__pt) noexcept`
- `template<typename _Up >`  
`constexpr _Up & get_underlying (propagate\_const < _Up > &__pt) noexcept`

#### 4.845.1 Detailed Description

`template<typename _Tp>`  
`class std::experimental::fundamentals_v2::propagate_const< _Tp >`

Const-propagating wrapper.

Definition at line 64 of file `propagate_const`.

The documentation for this class was generated from the following file:

- [propagate\\_const](#)

#### 4.846 `__gnu_pbds::quadratic_probe_fn< Size_Type >` Class Template Reference

##### Public Types

- `typedef Size_Type size_type`

##### Public Member Functions

- `void swap (quadratic\_probe\_fn < Size_Type > &other)`

## Protected Member Functions

- `size_type operator() (size_type i) const`

### 4.846.1 Detailed Description

```
template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >
```

A probe sequence policy using square increments.

Definition at line 85 of file hash\_policy.hpp.

### 4.846.2 Member Function Documentation

#### 4.846.2.1 operator()()

```
template<typename Size_Type >
quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value.

Definition at line 53 of file hash\_policy.hpp.

The documentation for this class was generated from the following file:

- [hash\\_policy.hpp](#)

## 4.847 std::queue< \_Tp, \_Sequence > Class Template Reference

### Public Types

- `typedef _Sequence::const_reference const_reference`
- `typedef _Sequence container_type`
- `typedef _Sequence::reference reference`
- `typedef _Sequence::size_type size_type`
- `typedef _Sequence::value_type value_type`

## Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>  
queue ()`
- `queue (const _Sequence &__c)`
- `queue (_Sequence &&__c)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>  
queue (const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>  
queue (const _Sequence &__c, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>  
queue (_Sequence &&__c, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>  
queue (const queue &__q, const _Alloc &__a)`
- `template<typename _Alloc , typename _Requires = _Uses<_Alloc>>  
queue (queue &&__q, const _Alloc &__a)`
- `reference back ()`
- `const_reference back () const`
- `template<typename... _Args>  
void emplace (_Args &&... __args)`
- `bool empty () const`
- `reference front ()`
- `const_reference front () const`
- `void pop ()`
- `void push (const value_type &__x)`
- `void push (value_type &&__x)`
- `size_type size () const`
- `void swap (queue &__q) noexcept(__is_nothrow_swappable< _Sequence >::value)`

## Protected Attributes

- `_Sequence c`

## Friends

- `template<typename _Tp1 , typename _Seq1 >  
bool operator< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`
- `template<typename _Tp1 , typename _Seq1 >  
bool operator== (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)`

## 4.847.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >
```

A standard container giving FIFO behavior.

### Template Parameters

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                         |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as `std::list` or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

Definition at line 96 of file `stl_queue.h`.

## 4.847.2 Constructor & Destructor Documentation

### 4.847.2.1 `queue()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::queue< _Tp, _Sequence >::queue () [inline]
```

Default constructor creates no elements.

Definition at line 166 of file `stl_queue.h`.

## 4.847.3 Member Function Documentation

### 4.847.3.1 `back()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::back () [inline]
```

Returns a read/write reference to the data at the last element of the queue.

Definition at line 238 of file `stl_queue.h`.

#### 4.847.3.2 `back()` [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::back () const [inline]
```

Returns a read-only (constant) reference to the data at the last element of the queue.

Definition at line 249 of file `stl_queue.h`.

#### 4.847.3.3 `empty()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue<_Tp, _Sequence>::empty () const [inline]
```

Returns true if the queue is empty.

Definition at line 203 of file `stl_queue.h`.

References `std::queue<_Tp, _Sequence>::c`.

#### 4.847.3.4 `front()` [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue<_Tp, _Sequence>::front () [inline]
```

Returns a read/write reference to the data at the first element of the queue.

Definition at line 216 of file `stl_queue.h`.

#### 4.847.3.5 `front()` [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue<_Tp, _Sequence>::front () const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the queue.

Definition at line 227 of file `stl_queue.h`.

#### 4.847.3.6 pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop () [inline]
```

Removes first element.

This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

Definition at line 298 of file stl\_queue.h.


#### 4.847.3.7 push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (
 const value_type & __x) [inline]
```

Add data to the end of the queue.



## Parameters

|                                                                                   |                   |
|-----------------------------------------------------------------------------------|-------------------|
|  | Data to be added. |
| <code>_X</code>                                                                   |                   |

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 265 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.

## 4.847.3.8 size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const [inline]
```

Returns the number of elements in the queue.

Definition at line 208 of file `stl_queue.h`.

References `std::queue< _Tp, _Sequence >::c`.


## 4.847.4 Member Data Documentation

## 4.847.4.1 c

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue< _Tp, _Sequence >::c [protected]
```

`c` is the underlying container.

Definition at line 153 of file `stl_queue.h`.

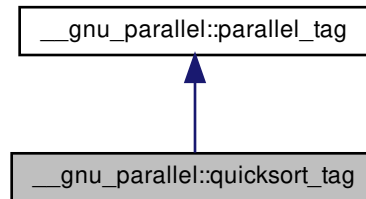
Referenced by `std::queue< _Tp, _Sequence >::empty()`, `std::operator<()`, `std::operator==()`, `std::queue< _Tp,  Sequence >::push()`, and `std::queue< _Tp, _Sequence >::size()`.

The documentation for this class was generated from the following file:

- [stl\\_queue.h](#)

#### 4.848 \_\_gnu\_parallel::quicksort\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::quicksort\_tag:



##### Public Member Functions

- **quicksort\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

##### 4.848.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

Definition at line 155 of file tags.h.

##### 4.848.2 Member Function Documentation

###### 4.848.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

###### 4.848.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

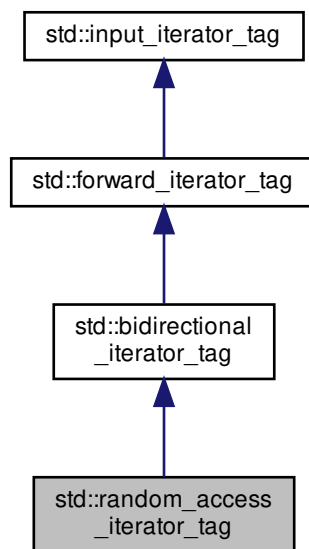
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.849 std::random\_access\_iterator\_tag Struct Reference

Inheritance diagram for std::random\_access\_iterator\_tag:



##### 4.849.1 Detailed Description

Random-access iterators support a superset of bidirectional iterator operations.

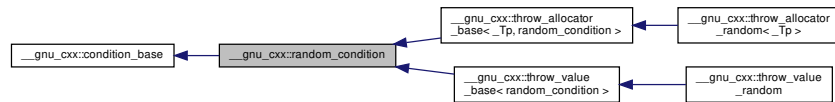
Definition at line 107 of file stl\_iterator\_base\_types.h.

The documentation for this struct was generated from the following file:

- [stl\\_iterator\\_base\\_types.h](#)

## 4.850 `__gnu_cxx::random_condition` Struct Reference

Inheritance diagram for `__gnu_cxx::random_condition`:



### Classes

- struct [always\\_adjustor](#)
- struct [group\\_adjustor](#)
- struct [never\\_adjustor](#)

### Public Member Functions

- void **seed** (unsigned long \_\_s)

### Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

#### 4.850.1 Detailed Description

Base class for random probability control and throw.

Definition at line 500 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.851 `std::random_device` Class Reference

### Public Types

- typedef unsigned int [result\\_type](#)

### Public Member Functions

- **random\_device** (const [std::string](#) &\_\_token)
- **random\_device** (const [random\\_device](#) &)=delete
- double **entropy** () const noexcept
- [result\\_type](#) **operator()** ()
- void **operator=** (const [random\\_device](#) &)=delete

### Static Public Member Functions

- static constexpr [result\\_type](#) **max** ()
- static constexpr [result\\_type](#) **min** ()

#### 4.851.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Definition at line 1608 of file `random.h`.

#### 4.851.2 Member Typedef Documentation

##### 4.851.2.1 `result_type`

```
typedef unsigned int std::random_device::result_type
```

The type of the generated random value.

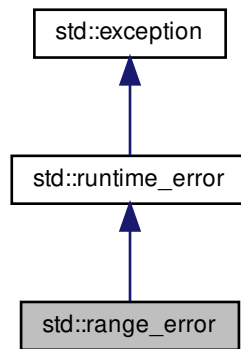
Definition at line 1612 of file `random.h`.

The documentation for this class was generated from the following file:

- [random.h](#)

## 4.852 std::range\_error Class Reference

Inheritance diagram for std::range\_error:



### Public Member Functions

- **range\_error** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **range\_error** (const char \*) \_GLIBCXX\_TXN\_SAFE
- **range\_error** (const [range\\_error](#) &)=default
- **range\_error** ([range\\_error](#) &&)=default
- [range\\_error](#) & **operator=** (const [range\\_error](#) &)=default
- [range\\_error](#) & **operator=** ([range\\_error](#) &&)=default
- virtual const char \* **what** () const noexcept

### 4.852.1 Detailed Description

Thrown to indicate range errors in internal computations.

Definition at line 258 of file stdexcept.

### 4.852.2 Member Function Documentation

4.852.2.1 `what()`

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

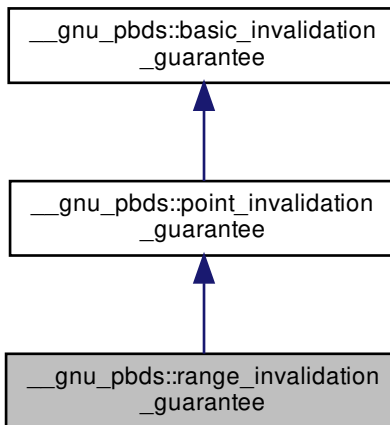
Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.853 `__gnu_pbds::range_invalidation_guarantee` Struct Reference

Inheritance diagram for `__gnu_pbds::range_invalidation_guarantee`:



## 4.853.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of `begin()` and `end()`) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.

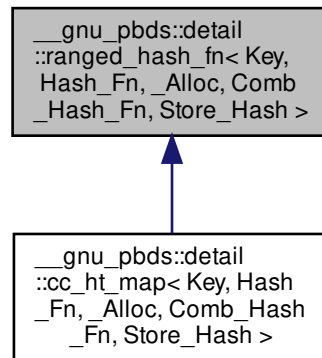
Definition at line 114 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.854 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >`:



##### 4.854.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >
```

Primary template.

Definition at line 56 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.855 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

##### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**



#### Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void `notify_resized` (size\_type)
- size\_type `operator()` (key\_const\_reference) const
- void `swap` (`ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >` &)

#### 4.855.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
```

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.

Definition at line 72 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.856 `__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, and `Comb_Hash_Fn`.

#### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**
- typedef `_Alloc::size_type` **size\_type**

#### Protected Member Functions

- `ranged_hash_fn` (size\_type)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &)
- `ranged_hash_fn` (size\_type, const Hash\_Fn &, const Comb\_Hash\_Fn &)
- void `notify_resized` (size\_type)
- `comp_hash operator()` (key\_const\_reference) const
- `comp_hash operator()` (key\_const\_reference, size\_type) const
- void `swap` (`ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >` &)

#### 4.856.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
```

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.

Definition at line 154 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

#### 4.857 \_\_gnu\_pbds::detail::ranged\_hash\_fn< Key, null\_type, \_Alloc, Comb\_Hash\_Fn, false > Class Template Reference

Inherits `Comb_Hash_Fn`.

##### Protected Types

- typedef `Comb_Hash_Fn` **comb\_hash\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

##### Protected Member Functions

- **ranged\_hash\_fn** (`size_type`)
- **ranged\_hash\_fn** (`size_type`, `const Comb_Hash_Fn &`)
- **ranged\_hash\_fn** (`size_type`, `const null_type &`, `const Comb_Hash_Fn &`)
- void **swap** ([ranged\\_hash\\_fn](#)< `Key`, `null_type`, `_Alloc`, `Comb_Hash_Fn`, `false` > &)

#### 4.857.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
```

Specialization 3 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values not be stored.

Definition at line 256 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 4.858 `__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >` Class Template Reference

Inherits `Comb_Hash_Fn`.

### Protected Types

- typedef `Comb_Hash_Fn` **`comb_hash_fn_base`**
- typedef `_Alloc::size_type` **`size_type`**

### Protected Member Functions

- **`ranged_hash_fn`** (`size_type`)
- **`ranged_hash_fn`** (`size_type`, `const Comb_Hash_Fn &`)
- **`ranged_hash_fn`** (`size_type`, `const null_type &`, `const Comb_Hash_Fn &`)
- void **`swap`** (`ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > &`)

### 4.858.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >
```

Specialization 4 The client does not supply a hash function (by specifying `null_type` as the `Hash_Fn` parameter), and requests that hash values be stored.

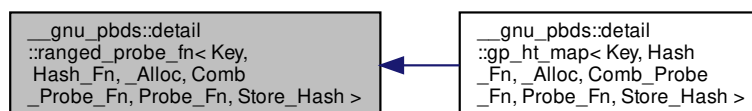
Definition at line 313 of file `ranged_hash_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_hash\\_fn.hpp](#)

## 4.859 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >`:



#### 4.859.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >
```

Primary template.

Definition at line 55 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

#### 4.860 \_\_gnu\_pbds::detail::ranged\_probe\_fn< Key, Hash\_Fn, \_Alloc, Comb\_Probe\_Fn, Probe\_Fn, false > Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

##### Protected Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn\_base**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef [rebind\\_traits](#)< `_Alloc`, `Key` >::const\_reference **key\_const\_reference**
- typedef `Probe_Fn` **probe\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

##### Protected Member Functions

- **ranged\_probe\_fn** (`size_type`)
- **ranged\_probe\_fn** (`size_type`, const `Hash_Fn` &)
- **ranged\_probe\_fn** (`size_type`, const `Hash_Fn` &, const `Comb_Probe_Fn` &)
- **ranged\_probe\_fn** (`size_type`, const `Hash_Fn` &, const `Comb_Probe_Fn` &, const `Probe_Fn` &)
- void **notify\_resized** (`size_type`)
- `size_type` **operator()** (`key_const_reference`) const
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) const
- void **swap** ([ranged\\_probe\\_fn](#)< `Key`, `Hash_Fn`, `_Alloc`, `Comb_Probe_Fn`, `Probe_Fn`, false > &)

#### 4.860.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
```

##### Specialization 1

The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 71 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

4.861 `__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >` Class Template Reference

Inherits `Hash_Fn`, `Comb_Probe_Fn`, and `Probe_Fn`.

## Protected Types

- typedef `Comb_Probe_Fn` **comb\_probe\_fn\_base**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `Hash_Fn` **hash\_fn\_base**
- typedef `rebind_traits< _Alloc, Key >::const_reference` **key\_const\_reference**
- typedef `Probe_Fn` **probe\_fn\_base**
- typedef `_Alloc::size_type` **size\_type**

## Protected Member Functions

- **ranged\_probe\_fn** (`size_type`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`)
- **ranged\_probe\_fn** (`size_type`, `const Hash_Fn &`, `const Comb_Probe_Fn &`, `const Probe_Fn &`)
- void **notify\_resized** (`size_type`)
- **comp\_hash operator()** (`key_const_reference`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`, `size_type`) `const`
- `size_type` **operator()** (`key_const_reference`, `size_type`) `const`
- void **swap** (`ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &`)

## 4.861.1 Detailed Description

```
template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
```

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.

Definition at line 176 of file `ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

4.862 `__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >` Class Template Reference

Inherits `Comb_Probe_Fn`.

## Protected Types

- typedef Comb\_Probe\_Fn **comb\_probe\_fn\_base**
- typedef [rebind\\_traits](#)<\_Alloc, Key >::const\_reference **key\_const\_reference**
- typedef \_Alloc::size\_type **size\_type**

## Protected Member Functions

- **ranged\_probe\_fn** (size\_type size)
- **ranged\_probe\_fn** (size\_type, const Comb\_Probe\_Fn &r\_comb\_probe\_fn)
- **ranged\_probe\_fn** (size\_type, const [null\\_type](#) &, const Comb\_Probe\_Fn &r\_comb\_probe\_fn, const [null\\_type](#) &)
- void **swap** ([ranged\\_probe\\_fn](#) &other)

## 4.862.1 Detailed Description

```
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >
```

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.

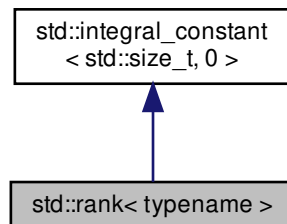
Definition at line 296 of file [ranged\\_probe\\_fn.hpp](#).

The documentation for this class was generated from the following file:

- [ranged\\_probe\\_fn.hpp](#)

## 4.863 std::rank&lt; typename &gt; Struct Template Reference

Inheritance diagram for std::rank< typename >:



#### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr std::size\_t **value**

#### 4.863.1 Detailed Description

```
template<typename>
struct std::rank< typename >
```

rank

Definition at line 1359 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.864 `std::ratio< _Num, _Den >` Struct Template Reference

#### Public Types

- typedef [ratio](#)< num, den > **type**

#### Static Public Attributes

- static constexpr intmax\_t **den**
- static constexpr intmax\_t **num**

#### 4.864.1 Detailed Description

```
template<intmax_t _Num, intmax_t _Den = 1>
struct std::ratio< _Num, _Den >
```

Provides compile-time rational arithmetic.

This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type `intmax_t`. The ratio is simplified when instantiated.

For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

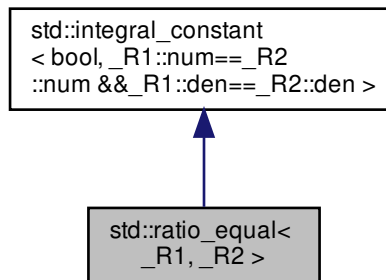
Definition at line 266 of file `ratio`.

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 4.865 `std::ratio_equal< _R1, _R2 >` Struct Template Reference

Inheritance diagram for `std::ratio_equal< _R1, _R2 >`:



#### Public Types

- typedef `integral_constant< bool, __v >` **type**
- typedef `bool` **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept



## Static Public Attributes

- static constexpr bool **value**

## 4.865.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_equal< _R1, _R2 >
```

ratio\_equal

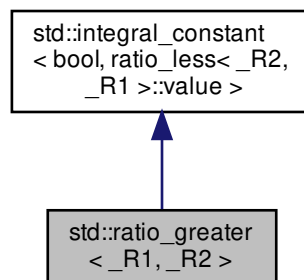
Definition at line 351 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

## 4.866 std::ratio\_greater&lt; \_R1, \_R2 &gt; Struct Template Reference

Inheritance diagram for std::ratio\_greater< \_R1, \_R2 >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 4.866.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater< _R1, _R2 >
```

ratio\_greater

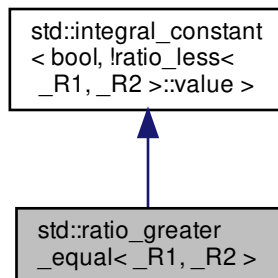
Definition at line 409 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 4.867 std::ratio\_greater\_equal< \_R1, \_R2 > Struct Template Reference

Inheritance diagram for std::ratio\_greater\_equal< \_R1, \_R2 >:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.867.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_greater_equal< _R1, _R2 >
```

ratio\_greater\_equal

Definition at line 415 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

4.868 `std::ratio_less< _R1, _R2 >` Struct Template Reference

Inherits type< \_R1, \_R2 >.

## 4.868.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_less< _R1, _R2 >
```

ratio\_less

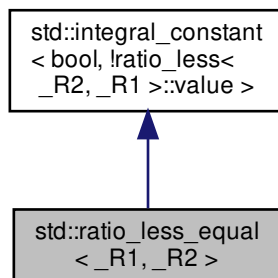
Definition at line 397 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

4.869 `std::ratio_less_equal< _R1, _R2 >` Struct Template Reference

Inheritance diagram for `std::ratio_less_equal< _R1, _R2 >`:



#### Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr bool **value**

#### 4.869.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_less_equal< _R1, _R2 >
```

ratio\_less\_equal

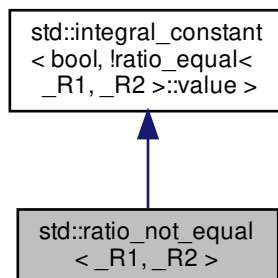
Definition at line 403 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

#### 4.870 std::ratio\_not\_equal< \_R1, \_R2 > Struct Template Reference

Inheritance diagram for std::ratio\_not\_equal< \_R1, \_R2 >:



## Public Types

- typedef [integral\\_constant](#)< bool, \_\_v > **type**
- typedef bool **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr bool **value**

## 4.870.1 Detailed Description

```
template<typename _R1, typename _R2>
struct std::ratio_not_equal< _R1, _R2 >
```

ratio\_not\_equal

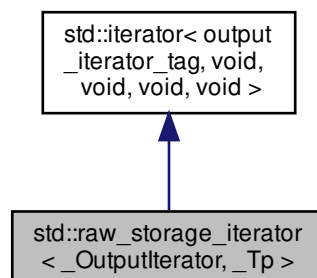
Definition at line 357 of file ratio.

The documentation for this struct was generated from the following file:

- [ratio](#)

4.871 `std::raw_storage_iterator< _OutputIterator, _Tp >` Class Template Reference

Inheritance diagram for `std::raw_storage_iterator< _OutputIterator, _Tp >`:



### Public Types

- typedef void [difference\\_type](#)
- typedef [output\\_iterator\\_tag](#) [iterator\\_category](#)
- typedef void [pointer](#)
- typedef void [reference](#)
- typedef void [value\\_type](#)

### Public Member Functions

- **raw\_storage\_iterator** ([\\_OutputIterator](#) \_\_x)
- [\\_OutputIterator](#) **base** () const
- [raw\\_storage\\_iterator](#) & **operator\*** ()
- [raw\\_storage\\_iterator](#) & **operator++** ()
- [raw\\_storage\\_iterator](#) **operator++** (int)
- [raw\\_storage\\_iterator](#) & **operator=** (const [\\_Tp](#) &\_\_element)
- [raw\\_storage\\_iterator](#) & **operator=** ([\\_Tp](#) &&\_\_element)

### Protected Attributes

- [\\_OutputIterator](#) **\_M\_iter**

#### 4.871.1 Detailed Description

```
template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >
```

This iterator class lets algorithms store their results into uninitialized memory.

Definition at line 68 of file `stl_raw_storage_iter.h`.

#### 4.871.2 Member Typedef Documentation

##### 4.871.2.1 [difference\\_type](#)

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::difference_type
[inherited]
```

Distance between iterators is represented as this type.

Definition at line 134 of file `stl_iterator_base_types.h`.

#### 4.871.2.2 `iterator_category`

```
typedef output_iterator_tag std::iterator< output_iterator_tag , void , void , void , void >↵
::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file `stl_iterator_base_types.h`.

#### 4.871.2.3 `pointer`

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::pointer [inherited]
```

This type represents a pointer-to-value\_type.

Definition at line 136 of file `stl_iterator_base_types.h`.

#### 4.871.2.4 `reference`

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::reference [inherited]
```

This type represents a reference-to-value\_type.

Definition at line 138 of file `stl_iterator_base_types.h`.

#### 4.871.2.5 `value_type`

```
typedef void std::iterator< output_iterator_tag , void , void , void , void >::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

The documentation for this class was generated from the following file:

- [stl\\_raw\\_storage\\_iter.h](#)

### 4.872 `__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >` Struct Template Reference

Inherits `std::_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >`.

## Public Types

- `template<typename _Iter >`  
`using __same_value_type = is_same< value_type, typename iterator_traits< _Iter >::value_type >`
- `typedef std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > _Base`
- `typedef _Base::allocator_type allocator_type`
- `typedef _Rb_tree_const_iterator< value_type > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::reverse\_iterator< const_iterator > const_reverse_iterator`
- `typedef ptrdiff_t difference_type`
- `typedef _Rb_tree_iterator< value_type > iterator`
- `typedef _Key key_type`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::reverse\_iterator< iterator > reverse_iterator`
- `typedef size_t size_type`
- `typedef _Val value_type`

## Public Member Functions

- `rb_tree (const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())`
- `bool __rb_verify () const`
- `template<typename _Iterator >`  
`void _M_assign_equal (_Iterator, _Iterator)`
- `template<typename _Iterator >`  
`void _M_assign_unique (_Iterator, _Iterator)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`size_type _M_count_tr (const _Kt &__k) const`
- `template<typename... _Args>`  
`iterator _M_emplace_equal (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_equal (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`iterator _M_emplace_hint_unique (const_iterator __pos, _Args &&... __args)`
- `template<typename... _Args>`  
`pair< iterator, bool > _M_emplace_unique (_Args &&... __args)`
- `template<typename... _Args>`  
`pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > _M_emplace_←  
unique (_Args &&... __args)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`pair< iterator, iterator > _M_equal_range_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`pair< const_iterator, const_iterator > _M_equal_range_tr (const _Kt &__k) const`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`iterator _M_find_tr (const _Kt &__k)`
- `template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>`  
`const_iterator _M_find_tr (const _Kt &__k) const`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_equal_pos (const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_equal_pos (const_iterator __pos, const key_type &__k)`
- `pair< _Base_ptr, _Base_ptr > _M_get_insert_hint_unique_pos (const_iterator __pos, const key_type &__k)`



- `pair< _Base_ptr, _Base_ptr > _M_get_insert_unique_pos` (const key\_type &\_\_k)
- `_Node_allocator & _M_get_Node_allocator` () noexcept
- `const _Node_allocator & _M_get_Node_allocator` () const noexcept
- `template<typename _Arg >`  
`iterator _M_insert_equal` (\_Arg &&\_\_x)
- `template<typename _Arg, typename _NodeGen >`  
`iterator _M_insert_equal` (const\_iterator \_\_pos, \_Arg &&\_\_x, \_NodeGen &)
- `template<typename _Arg >`  
`iterator _M_insert_equal` (const\_iterator \_\_pos, \_Arg &&\_\_x)
- `template<typename _InputIterator >`  
`__enable_if_t< __same_value_type< _InputIterator >::value > _M_insert_range_equal` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `template<typename _InputIterator >`  
`__enable_if_t<! __same_value_type< _InputIterator >::value > _M_insert_range_equal` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `template<typename _InputIterator >`  
`__enable_if_t< __same_value_type< _InputIterator >::value > _M_insert_range_unique` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `template<typename _InputIterator >`  
`__enable_if_t<! __same_value_type< _InputIterator >::value > _M_insert_range_unique` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `template<typename _Arg >`  
`pair< iterator, bool > _M_insert_unique` (\_Arg &&\_\_x)
- `template<typename _Arg >`  
`pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > _M_insert_unique` (\_Arg &&\_\_v)
- `template<typename _Arg, typename _NodeGen >`  
`iterator _M_insert_unique` (const\_iterator \_\_pos, \_Arg &&\_\_x, \_NodeGen &)
- `template<typename _Arg >`  
`iterator _M_insert_unique` (const\_iterator \_\_pos, \_Arg &&\_\_x)
- `template<typename _Kt, typename _Req = __has_is_transparent_t< _Compare, _Kt >>`  
`iterator _M_lower_bound_tr` (const \_Kt &\_\_k)
- `template<typename _Kt, typename _Req = __has_is_transparent_t< _Compare, _Kt >>`  
`const_iterator _M_lower_bound_tr` (const \_Kt &\_\_k) const
- `template<typename _Kt, typename _Req = __has_is_transparent_t< _Compare, _Kt >>`  
`iterator _M_upper_bound_tr` (const \_Kt &\_\_k)
- `template<typename _Kt, typename _Req = __has_is_transparent_t< _Compare, _Kt >>`  
`const_iterator _M_upper_bound_tr` (const \_Kt &\_\_k) const
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `void clear` () noexcept
- `size_type count` (const key\_type &\_\_k) const
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `pair< iterator, iterator > equal_range` (const key\_type &\_\_k)
- `pair< const_iterator, const_iterator > equal_range` (const key\_type &\_\_k) const
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const\_iterator \_\_position)
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (iterator \_\_position)
- `size_type erase` (const key\_type &\_\_x)
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const\_iterator \_\_first, const\_iterator \_\_last)
- `iterator find` (const key\_type &\_\_k)

- const\_iterator **find** (const key\_type &\_\_k) const
- allocator\_type **get\_allocator** () const noexcept
- \_Compare **key\_comp** () const
- iterator **lower\_bound** (const key\_type &\_\_k)
- const\_iterator **lower\_bound** (const key\_type &\_\_k) const
- size\_type **max\_size** () const noexcept
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- size\_type **size** () const noexcept
- void **swap** (\_Rb\_tree &\_\_t) noexcept(*/\*conditional \*/*)
- iterator **upper\_bound** (const key\_type &\_\_k)
- const\_iterator **upper\_bound** (const key\_type &\_\_k) const

#### Protected Types

- typedef \_Rb\_tree\_node\_base \* **\_Base\_ptr**
- typedef const \_Rb\_tree\_node\_base \* **\_Const\_Base\_ptr**
- typedef const \_Rb\_tree\_node< \_Val > \* **\_Const\_Link\_type**
- typedef \_Rb\_tree\_node< \_Val > \* **\_Link\_type**

#### Protected Member Functions

- \_Link\_type **\_M\_begin** () noexcept
- \_Const\_Link\_type **\_M\_begin** () const noexcept
- template<typename \_NodeGen >  
\_Link\_type **\_M\_clone\_node** (\_Const\_Link\_type \_\_x, \_NodeGen &\_\_node\_gen)
- template<typename... \_Args>  
void **\_M\_construct\_node** (\_Link\_type \_\_node, \_Args &&... \_\_args)
- template<typename... \_Args>  
\_Link\_type **\_M\_create\_node** (\_Args &&... \_\_args)
- void **\_M\_destroy\_node** (\_Link\_type \_\_p) noexcept
- void **\_M\_drop\_node** (\_Link\_type \_\_p) noexcept
- \_Base\_ptr **\_M\_end** () noexcept
- \_Const\_Base\_ptr **\_M\_end** () const noexcept
- \_Link\_type **\_M\_get\_node** ()
- \_Base\_ptr & **\_M\_leftmost** () noexcept
- \_Const\_Base\_ptr **\_M\_leftmost** () const noexcept
- void **\_M\_put\_node** (\_Link\_type \_\_p) noexcept
- \_Base\_ptr & **\_M\_rightmost** () noexcept
- \_Const\_Base\_ptr **\_M\_rightmost** () const noexcept
- \_Base\_ptr & **\_M\_root** () noexcept
- \_Const\_Base\_ptr **\_M\_root** () const noexcept

#### Static Protected Member Functions

- `static const _Key & _S_key ( _Const_Link_type __x)`
- `static const _Key & _S_key ( _Const_Base_ptr __x)`
- `static _Link_type _S_left ( _Base_ptr __x) noexcept`
- `static _Const_Link_type _S_left ( _Const_Base_ptr __x) noexcept`
- `static _Base_ptr _S_maximum ( _Base_ptr __x) noexcept`
- `static _Const_Base_ptr _S_maximum ( _Const_Base_ptr __x) noexcept`
- `static _Base_ptr _S_minimum ( _Base_ptr __x) noexcept`
- `static _Const_Base_ptr _S_minimum ( _Const_Base_ptr __x) noexcept`
- `static _Link_type _S_right ( _Base_ptr __x) noexcept`
- `static _Const_Link_type _S_right ( _Const_Base_ptr __x) noexcept`

#### Protected Attributes

- `_Rb_tree_impl< _Compare > _M_impl`

#### 4.872.1 Detailed Description

```
template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = std::allocator<_Value>>
struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

Definition at line 77 of file `rb_tree`.

The documentation for this struct was generated from the following file:

- [rb\\_tree](#)

#### 4.873 `__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class` Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `rb_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**
- typedef `base_type::node_update` **node\_update**
- typedef `base_type::const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `base_type::pointer` **pointer**
- typedef `base_type::reference` **reference**
- typedef `base_type::reverse_iterator` **reverse\_iterator**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `base_type::value_type` **value\_type**

## Public Member Functions

- **rb\_tree\_map** (const `Cmp_Fn` &)
- **rb\_tree\_map** (const `Cmp_Fn` &, const `node_update` &)
- **rb\_tree\_map** (const `rb_tree_map`< `Key`, `Mapped`, `Cmp_Fn`, `Node_And_It_Traits`, `_Alloc` > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename `It` >  
void **copy\_from\_range** (`It`, `It`)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (`key_const_reference`)

- iterator **erase** (iterator)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- `std::pair`< point\_iterator, bool > **insert** (const\_reference)
- void **join** (`rb_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, `rb_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (`rb_tree_map`< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

#### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

#### Protected Types

- typedef node\_alloc\_traits::value\_type **node**
- typedef node\_alloc\_traits::allocator\_type **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef `types_traits`< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- **std::pair**< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- **std::pair**< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

### Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

#### 4.873.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Red-Black tree.

This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

Definition at line 84 of file rb\_tree\_.hpp.

## 4.873.2 Member Function Documentation

### 4.873.2.1 `node_begin()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node↵
begin () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to the node at the root of the tree.

Definition at line 111 of file `bin_search_tree.hpp`.

### 4.873.2.2 `node_begin()` [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node↵
begin () [inline], [inherited]
```

Returns a `node_iterator` corresponding to the node at the root of the tree.

Definition at line 119 of file `bin_search_tree.hpp`.

### 4.873.2.3 `node_end()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node↵
end () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 127 of file `bin_search_tree.hpp`.

#### 4.873.2.4 node\_end() [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_←
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end (
) [inline], [inherited]
```

Returns a node\_iterator corresponding to a node just after a leaf of the tree.

Definition at line 135 of file bin\_search\_tree\_.hpp.

The documentation for this class was generated from the following file:

- [rb\\_tree\\_.hpp](#)

#### 4.874 \_\_gnu\_pbds::detail::rb\_tree\_node\_< Value\_Type, Metadata, \_Alloc > Struct Template Reference

##### Public Types

- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::const\_reference **metadata\_const\_reference**
- typedef [rebind\\_traits](#)< \_Alloc, metadata\_type >::reference **metadata\_reference**
- typedef Metadata **metadata\_type**
- typedef [rebind\\_traits](#)< \_Alloc, [rb\\_tree\\_node\\_](#) >::pointer **node\_pointer**
- typedef Value\_Type **value\_type**

##### Public Member Functions

- metadata\_const\_reference **get\_metadata** () const
- metadata\_reference **get\_metadata** ()
- bool **special** () const

##### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_red**
- value\_type **m\_value**



## 4.874.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >
```

Node for Red-Black trees.

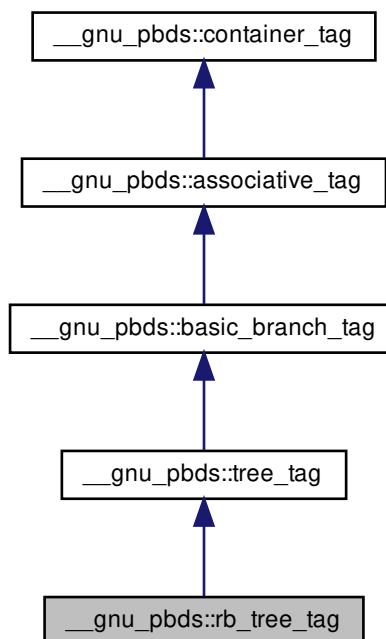
Definition at line 52 of file `rb_tree_map_/node.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/node.hpp](#)

## 4.875 \_\_gnu\_pbds::rb\_tree\_tag Struct Reference

Inheritance diagram for `__gnu_pbds::rb_tree_tag`:



## 4.875.1 Detailed Description

Red-black tree.

Definition at line 153 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.876 `__gnu_pbds::detail::rc< _Node, _Alloc >` Class Template Reference

##### Public Types

- typedef entry\_const\_pointer **const\_iterator**
- typedef node\_pointer **entry**

##### Public Member Functions

- **rc** (const [rc](#) &)
- const const\_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const\_iterator **end** () const
- void **pop** ()
- void **push** (entry)
- size\_type **size** () const
- void **swap** ([rc](#) &)
- node\_pointer **top** () const

##### 4.876.1 Detailed Description

```
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >
```

Redundant binary counter.

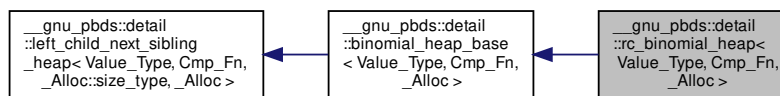
Definition at line 50 of file rc.hpp.

The documentation for this class was generated from the following file:

- [rc.hpp](#)

#### 4.877 `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >`:



## Public Types

- `typedef base_type::allocator_type allocator_type`
- `typedef base_type::cmp_fn cmp_fn`
- `typedef base_type::const_iterator const_iterator`
- `typedef base_type::const_pointer const_pointer`
- `typedef base_type::const_reference const_reference`
- `typedef _Alloc::difference_type difference_type`
- `typedef base_type::iterator iterator`
- `typedef base_type::point_const_iterator point_const_iterator`
- `typedef base_type::point_iterator point_iterator`
- `typedef base_type::pointer pointer`
- `typedef base_type::reference reference`
- `typedef _Alloc::size_type size_type`
- `typedef Value_Type value_type`

## Public Member Functions

- `rc_binomial_heap` (const Cmp\_Fn &)
- `rc_binomial_heap` (const `rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- `iterator begin` ()
- `const_iterator begin` () const
- void `clear` ()
- bool `empty` () const
- `iterator end` ()
- `const_iterator end` () const
- void `erase` (`point_iterator`)
- `template<typename Pred >`  
`size_type erase_if` (Pred)
- Cmp\_Fn & `get_cmp_fn` ()
- const Cmp\_Fn & `get_cmp_fn` () const
- void `join` (`rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `join` (`binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- `size_type max_size` () const
- void `modify` (`point_iterator`, const\_reference)
- void `modify` (`point_iterator`, const\_reference)
- void `pop` ()
- `point_iterator push` (const\_reference)
- `point_iterator push` (const\_reference)
- `size_type size` () const
- `template<typename Pred >`  
void `split` (Pred, `rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- `template<typename Pred >`  
void `split` (Pred, `binomial_heap_base`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap` (`rc_binomial_heap`< Value\_Type, Cmp\_Fn, \_Alloc > &)
- void `swap` (`left_child_next_sibling_heap`< Value\_Type, Cmp\_Fn, \_Alloc::size\_type, \_Alloc > &)
- const\_reference `top` () const

### Protected Types

- typedef [base\\_type::node](#) **node**
- typedef `alloc_traits::allocator_type` **node\_allocator**
- typedef `_Alloc::size_type` **node\_metadata**
- typedef `std::pair`< `node_pointer`, `node_pointer` > **node\_pointer\_pair**

### Protected Member Functions

- void **actual\_erase\_node** (`node_pointer`)
- void **bubble\_to\_top** (`node_pointer`)
- void **clear\_imp** (`node_pointer`)
- template<typename It >  
void **copy\_from\_range** (It, It)
- void **find\_max** ()
- `node_pointer` **get\_new\_node\_for\_insert** (`const_reference`)
- `node_pointer` **prune** (`Pred`)
- void **swap** ([binomial\\_heap\\_base](#)< `Value_Type`, `Cmp_Fn`, `_Alloc` > &)
- void **swap\_with\_parent** (`node_pointer`, `node_pointer`)
- void **to\_linked\_list** ()
- void **value\_swap** ([left\\_child\\_next\\_sibling\\_heap](#) &)

### Static Protected Member Functions

- static void **make\_child\_of** (`node_pointer`, `node_pointer`)
- static `node_pointer` **parent** (`node_pointer`)

### Protected Attributes

- `node_pointer` **m\_p\_max**
- `node_pointer` **m\_p\_root**
- `size_type` **m\_size**

#### 4.877.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
```

Redundant-counter binomial heap.

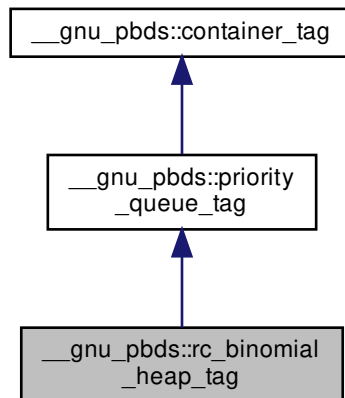
Definition at line 66 of file `rc_binomial_heap.hpp`.

The documentation for this class was generated from the following file:

- [rc\\_binomial\\_heap.hpp](#)

## 4.878 \_\_gnu\_pbds::rc\_binomial\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::rc\_binomial\_heap\_tag:



## 4.878.1 Detailed Description

Redundant-counter binomial-heap.

Definition at line 180 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.879 \_\_gnu\_pbds::detail::rebind\_traits&lt; \_Alloc, T &gt; Struct Template Reference

Inherits template `rebind_traits< T >`.

## Public Types

- using **const\_reference** = const T &
- using **reference** = T &

#### 4.879.1 Detailed Description

```
template<typename _Alloc, typename T>
struct __gnu_pbds::detail::rebind_traits< _Alloc, T >
```

Consistent API for accessing allocator-related types.

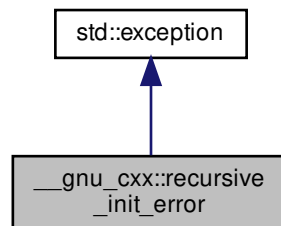
Definition at line 137 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.880 \_\_gnu\_cxx::recursive\_init\_error Class Reference

Inheritance diagram for \_\_gnu\_cxx::recursive\_init\_error:



#### Public Member Functions

- virtual const char \* [what](#) () const noexcept

#### 4.880.1 Detailed Description

Exception thrown by \_\_cxa\_guard\_acquire.

C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.

Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

Definition at line 696 of file cxxabi.h.

The documentation for this class was generated from the following file:

- [cxxabi.h](#)

## 4.881 `std::recursive_mutex` Class Reference

Inherits `std::__recursive_mutex_base`.

### Public Types

- `typedef __native_type * native_handle_type`

### Public Member Functions

- `recursive_mutex` (const [recursive\\_mutex](#) &)=delete
- `void lock` ()
- `native_handle_type native_handle` () noexcept
- [recursive\\_mutex](#) & `operator=` (const [recursive\\_mutex](#) &)=delete
- `bool try_lock` () noexcept
- `void unlock` ()

### Private Types

- `typedef __gthread_recursive_mutex_t __native_type`

### Private Attributes

- `__native_type _M_mutex`

#### 4.881.1 Detailed Description

The standard recursive mutex type.

Definition at line 92 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 4.882 `std::recursive_timed_mutex` Class Reference

### Public Member Functions

- `recursive_timed_mutex` (const [recursive\\_timed\\_mutex](#) &)=delete
- `void lock` ()
- [recursive\\_timed\\_mutex](#) & `operator=` (const [recursive\\_timed\\_mutex](#) &)=delete
- `bool try_lock` ()
- `template<typename _Rep, typename _Period >`  
`bool try_lock_for` (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- `template<typename _Clock, typename _Duration >`  
`bool try_lock_until` (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- `void unlock` ()

#### 4.882.1 Detailed Description

`recursive_timed_mutex`

Definition at line 408 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

#### 4.883 `std::bitset<_Nb>::reference` Class Reference

##### Public Member Functions

- **reference** ([bitset](#) &\_\_b, `size_t __pos`) `noexcept`
- **reference** (const [reference](#) &)=default
- [reference](#) & **flip** () `noexcept`
- **operator bool** () `const noexcept`
- [reference](#) & **operator=** (`bool __x`) `noexcept`
- [reference](#) & **operator=** (const [reference](#) &\_\_j) `noexcept`
- `bool` **operator~** () `const noexcept`

##### Friends

- class **bitset**

#### 4.883.1 Detailed Description

```
template<size_t _Nb>
class std::bitset<_Nb>::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from `bool` are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)

Definition at line 802 of file `bitset`.

The documentation for this class was generated from the following file:

- [bitset](#)



## 4.884 std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::reference Class Reference

### Public Member Functions

- **reference** ([dynamic\\_bitset](#) &\_\_b, size\_type \_\_pos) noexcept
- [reference](#) & **flip** () noexcept
- **operator bool** () const noexcept
- [reference](#) & **operator=** (bool \_\_x) noexcept
- [reference](#) & **operator=** (const [reference](#) &\_\_j) noexcept
- bool **operator~** () const noexcept

### Friends

- class **dynamic\_bitset**

### 4.884.1 Detailed Description

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference
```

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.

Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.

(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)

Definition at line 513 of file dynamic\_bitset.

The documentation for this class was generated from the following file:

- [dynamic\\_bitset](#)

## 4.885 std::reference\_wrapper< \_Tp > Class Template Reference

Inherits [\\_Reference\\_wrapper\\_base\\_memfun< remove\\_cv< \\_Tp >::type >](#).

### Public Types

- typedef [\\_Tp](#) **type**

## Public Member Functions

- `template<typename _Up, typename = __not_same<_Up>, typename = decltype(reference_wrapper::_S_fun(std::declval<_Up>()))>`  
`constexpr reference_wrapper (_Up &&__uref) noexcept(noexcept(reference_wrapper::_S_fun(std::declval<_Up>())`  
`↪ _Up >()))`
- `reference_wrapper (const reference_wrapper &)=default`
- `constexpr _Tp & get () const noexcept`
- `constexpr operator _Tp & () const noexcept`
- `template<typename... _Args>`  
`constexpr result_of<_Tp &(_Args &&...)>::type operator() (_Args &&... __args) const`
- `reference_wrapper & operator= (const reference_wrapper &)=default`

## Related Functions

(Note that these are not member functions.)

- `template<typename _Tp >`  
`constexpr reference_wrapper<_Tp> ref (_Tp &__t) noexcept`
- `template<typename _Tp >`  
`constexpr reference_wrapper<const _Tp> cref (const _Tp &__t) noexcept`
- `template<typename _Tp >`  
`constexpr reference_wrapper<_Tp> ref (reference_wrapper<_Tp> __t) noexcept`
- `template<typename _Tp >`  
`constexpr reference_wrapper<const _Tp> cref (reference_wrapper<_Tp> __t) noexcept`

### 4.885.1 Detailed Description

```
template<typename _Tp>
class std::reference_wrapper<_Tp>
```

Primary class template for `reference_wrapper`.

Definition at line 2160 of file `type_traits`.

### 4.885.2 Friends And Related Function Documentation

#### 4.885.2.1 `cref()` [1/2]

```
template<typename _Tp >
constexpr reference_wrapper<const _Tp> cref (
 const _Tp & __t) [related]
```

Denotes a const reference should be taken to a variable.

Definition at line 371 of file `refwrap.h`.

#### 4.885.2.2 cref() [2/2]

```
template<typename _Tp >
constexpr reference_wrapper< const _Tp > cref (
 reference_wrapper< _Tp > __t) [related]
```

std::cref overload to prevent wrapping a reference\_wrapper

Definition at line 391 of file refwrap.h.

#### 4.885.2.3 ref() [1/2]

```
template<typename _Tp >
constexpr reference_wrapper< _Tp > ref (
 _Tp & __t) [related]
```

Denotes a reference should be taken to a variable.

Definition at line 364 of file refwrap.h.

#### 4.885.2.4 ref() [2/2]

```
template<typename _Tp >
constexpr reference_wrapper< _Tp > ref (
 reference_wrapper< _Tp > __t) [related]
```

std::ref overload to prevent wrapping a reference\_wrapper

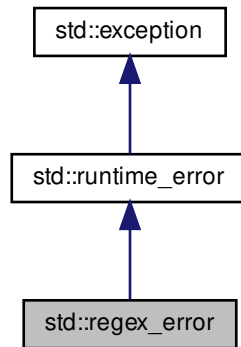
Definition at line 384 of file refwrap.h.

The documentation for this class was generated from the following files:

- [type\\_traits](#)
- [refwrap.h](#)

## 4.886 std::regex\_error Class Reference

Inheritance diagram for std::regex\_error:



### Public Member Functions

- [regex\\_error](#) ([regex\\_constants::error\\_type](#) \_\_ecode)
- [regex\\_constants::error\\_type](#) [code](#) () const
- virtual const char \* [what](#) () const noexcept

### Friends

- void [\\_\\_throw\\_regex\\_error](#) ([regex\\_constants::error\\_type](#), const char \*)

### 4.886.1 Detailed Description

A regular expression exception class.

The regular expression library throws objects of this class on error.

Definition at line 132 of file `regex_error.h`.

### 4.886.2 Constructor & Destructor Documentation

#### 4.886.2.1 regex\_error()

```
std::regex_error::regex_error (
 regex_constants::error_type __ecode) [explicit]
```

Constructs a `regex_error` object.

## Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__ecode</code> | the regex error code. |
|----------------------|-----------------------|

## 4.886.3 Member Function Documentation

4.886.3.1 `code()`

```
regex_constants::error_type std::regex_error::code () const [inline]
```

Gets the regex error code.

## Returns

the regex error code.

Definition at line 153 of file `regex_error.h`.

4.886.3.2 `what()`

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [regex\\_error.h](#)

4.887 `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits>` Class Template Reference

## Public Types

- typedef `std::ptrdiff_t` **difference\_type**
- typedef [std::forward\\_iterator\\_tag](#) **iterator\_category**
- typedef const [value\\_type](#) \* **pointer**
- typedef const [value\\_type](#) & **reference**
- typedef [basic\\_regex](#)< [\\_Ch\\_type](#), [\\_Rx\\_traits](#) > **regex\_type**
- typedef [match\\_results](#)< [\\_Bi\\_iter](#) > **value\_type**

## Public Member Functions

- `regex_iterator()`=default
- `regex_iterator` (`_Bi_iter __a`, `_Bi_iter __b`, const `regex_type &__re`, `regex_constants::match_flag_type __m`↵  
=`regex_constants::match_default`)
- `regex_iterator` (`_Bi_iter`, `_Bi_iter`, const `regex_type &&`, `regex_constants::match_flag_type=regex_constants::match_default`)=default
- `regex_iterator` (const `regex_iterator &`)=default
- bool `operator!=` (const `regex_iterator &__rhs`) const noexcept
- const `value_type & operator*` () const noexcept
- `regex_iterator & operator++` ()
- `regex_iterator operator++` (int)
- const `value_type * operator->` () const noexcept
- `regex_iterator & operator=` (const `regex_iterator &`)=default
- bool `operator==` (const `regex_iterator &`) const noexcept

## 4.887.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

An iterator adaptor that will provide repeated calls of `regex_search` over a range until no more matches remain.

Definition at line 2625 of file `regex.h`.

## 4.887.2 Constructor &amp; Destructor Documentation

4.887.2.1 `regex_iterator()` [1/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator () [default]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.

Referenced by `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`.

4.887.2.2 `regex_iterator()` [2/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a `regex_iterator`...

## Parameters

|                         |                                                    |
|-------------------------|----------------------------------------------------|
| <code>_↵<br/>_a</code>  | [IN] The start of a text range to search.          |
| <code>_↵<br/>_b</code>  | [IN] One-past-the-end of the text range to search. |
| <code>_↵<br/>_re</code> | [IN] The regular expression to match.              |
| <code>_↵<br/>_m</code>  | [IN] Policy flags for match rules.                 |

Definition at line 2648 of file regex.h.

References `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator()`, and `std::regex_search()`.

## 4.887.2.3 regex\_iterator() [3/3]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
 const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > &) [default]
```

Copy constructs a `regex_iterator`.

## 4.887.3 Member Function Documentation

## 4.887.3.1 operator!=(())

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!=(
 const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const [inline],
[noexcept]
```

Tests the inequivalence of two `regex_iterator`s.

Definition at line 2682 of file regex.h.

#### 4.887.3.2 operator\*()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* () const
[inline], [noexcept]
```

Dereferences a regex\_iterator.

Definition at line 2689 of file regex.h.

#### 4.887.3.3 operator++() [1/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()
```

Increments a regex\_iterator.

#### 4.887.3.4 operator++() [2/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
 int) [inline]
```

Postincrements a regex\_iterator.

Definition at line 2709 of file regex.h.

#### 4.887.3.5 operator->()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type* std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> () const
[inline], [noexcept]
```

Selects a regex\_iterator member.

Definition at line 2696 of file regex.h.



## 4.887.3.6 operator=()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator& std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
 const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > &) [default]
```

Copy assigns one regex\_iterator to another.

## 4.887.3.7 operator==()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
 const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > &) const [noexcept]
```

Tests the equivalence of two regex iterators.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.888 std::regex\_token\_iterator&lt; \_Bi\_iter, \_Ch\_type, \_Rx\_traits &gt; Class Template Reference

## Public Types

- typedef std::ptrdiff\_t **difference\_type**
- typedef std::forward\_iterator\_tag **iterator\_category**
- typedef const value\_type \* **pointer**
- typedef const value\_type & **reference**
- typedef basic\_regex< \_Ch\_type, \_Rx\_traits > **regex\_type**
- typedef sub\_match< \_Bi\_iter > **value\_type**

## Public Member Functions

- [regex\\_token\\_iterator](#) ()
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, int \_\_submatch=0, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const std::vector< int > &\_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- [regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, [initializer\\_list](#)< int > \_\_submatches, [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- template<std::size\_t \_Nm>  
[regex\\_token\\_iterator](#) ( \_Bi\_iter \_\_a, \_Bi\_iter \_\_b, const [regex\\_type](#) &\_\_re, const int(&\_\_submatches)[\_Nm], [regex\\_constants::match\\_flag\\_type](#) \_\_m=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, int=0, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))
- **regex\_token\_iterator** ( \_Bi\_iter, \_Bi\_iter, const [regex\\_type](#) &&, const std::vector< int > &, [regex\\_constants::match\\_flag\\_type](#)=[regex\\_constants::match\\_default](#))

- **regex\_token\_iterator** (*\_Bi\_iter*, *\_Bi\_iter*, const *regex\_type* &&, *initializer\_list*< int >, *regex\_constants::match\_flag\_type*=*regex\_constants::match\_default*) const
- *template*<*std::size\_t* *Nm*>  
**regex\_token\_iterator** (*\_Bi\_iter*, *\_Bi\_iter*, const *regex\_type* &&, const int(&)[*Nm*], *regex\_constants::match\_flag\_type*=*regex\_constants::match\_default*) const
- *regex\_token\_iterator* (const *regex\_token\_iterator* &\_\_rhs)
- bool *operator!=* (const *regex\_token\_iterator* &\_\_rhs) const
- const *value\_type* & *operator\** () const
- *regex\_token\_iterator* & *operator++* ()
- *regex\_token\_iterator* *operator++* (int)
- const *value\_type* \* *operator->* () const
- *regex\_token\_iterator* & *operator=* (const *regex\_token\_iterator* &\_\_rhs)
- bool *operator==* (const *regex\_token\_iterator* &\_\_rhs) const

#### 4.888.1 Detailed Description

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
```

Iterates over submatches in a range (or *splits* a text string).

The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a *std::sub\_match* object.

Definition at line 2742 of file *regex.h*.

#### 4.888.2 Constructor & Destructor Documentation

##### 4.888.2.1 *regex\_token\_iterator*() [1/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator () [inline]
```

Default constructs a *regex\_token\_iterator*.

A default-constructed *regex\_token\_iterator* is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Definition at line 2760 of file *regex.h*.

##### 4.888.2.2 *regex\_token\_iterator*() [2/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 int __submatch = 0,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a *regex\_token\_iterator*...

## Parameters

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__a</code>        | [IN] The start of the text to search.                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>__b</code>        | [IN] One-past-the-end of the text to search.                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>__re</code>       | [IN] The regular expression to search for.                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>__submatch</code> | [IN] Which submatch to return. There are some special values for this parameter: <ul style="list-style-type: none"> <li>• -1 each enumerated subexpression does NOT match the regular expression (aka field splitting)</li> <li>• 0 the entire string matching the subexpression is returned for each match within the text.</li> <li>• &gt;0 enumerates only the indicated subexpression from a match within the text.</li> </ul> |
| <code>__m</code>        | [IN] Policy flags for match rules.                                                                                                                                                                                                                                                                                                                                                                                                 |

Definition at line 2782 of file regex.h.

## 4.888.2.3 regex\_token\_iterator() [3/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 _Bi_iter __a,
 _Bi_iter __b,
 const regex_type & __re,
 const std::vector< int > & __submatches,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]
```

Constructs a regex\_token\_iterator...

## Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

Definition at line 2798 of file regex.h.

## 4.888.2.4 regex\_token\_iterator() [4/6]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
```

```

 __Bi_iter __a,
 __Bi_iter __b,
 const regex_type & __re,
 initializer_list< int > __submatches,
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]

```

Constructs a regex\_token\_iterator...

#### Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

Definition at line 2815 of file regex.h.

#### 4.888.2.5 regex\_token\_iterator() [5/6]

```

template<typename __Bi_iter , typename __Ch_type = typename iterator_traits<__Bi_iter>::value_type,
typename __Rx_traits = regex_traits<__Ch_type>>
template<std::size_t __Nm>
std::regex_token_iterator< __Bi_iter, __Ch_type, __Rx_traits >::regex_token_iterator (
 __Bi_iter __a,
 __Bi_iter __b,
 const regex_type & __re,
 const int (&) __submatches[__Nm],
 regex_constants::match_flag_type __m = regex_constants::match_default) [inline]

```

Constructs a regex\_token\_iterator...

#### Parameters

|                           |                                                                                            |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>__a</code>          | [IN] The start of the text to search.                                                      |
| <code>__b</code>          | [IN] One-past-the-end of the text to search.                                               |
| <code>__re</code>         | [IN] The regular expression to search for.                                                 |
| <code>__submatches</code> | [IN] A list of subexpressions to return for each regular expression match within the text. |
| <code>__m</code>          | [IN] Policy flags for match rules.                                                         |

Definition at line 2833 of file regex.h.

#### 4.888.2.6 regex\_token\_iterator() [6/6]

```

template<typename __Bi_iter , typename __Ch_type = typename iterator_traits<__Bi_iter>::value_type,
typename __Rx_traits = regex_traits<__Ch_type>>

```

```
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) [inline]
```

Copy constructs a regex\_token\_iterator.

#### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|

Definition at line 2865 of file regex.h.

### 4.888.3 Member Function Documentation

#### 4.888.3.1 operator!=(())

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator!= (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const [inline]
```

Compares a regex\_token\_iterator to another for inequality.

Definition at line 2887 of file regex.h.

#### 4.888.3.2 operator\*()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type& std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()
const [inline]
```

Dereferences a regex\_token\_iterator.

Definition at line 2894 of file regex.h.

#### 4.888.3.3 operator++() [1/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator& std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ ()
```

Increments a regex\_token\_iterator.

#### 4.888.3.4 operator++() [2/2]

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
 int) [inline]
```

Postincrements a regex\_token\_iterator.

Definition at line 2914 of file regex.h.

#### 4.888.3.5 operator->()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type* std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()
const [inline]
```

Selects a regex\_token\_iterator member.

Definition at line 2901 of file regex.h.

#### 4.888.3.6 operator=()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator& std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Assigns a regex\_token\_iterator to another.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <code>__rhs</code> | [IN] A regex_token_iterator to copy. |
|--------------------|--------------------------------------|

#### 4.888.3.7 operator==()

```
template<typename _Bi_iter , typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
 const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const
```

Compares a regex\_token\_iterator to another for equality.

The documentation for this class was generated from the following file:

- [regex.h](#)

## 4.889 std::regex\_traits&lt;\_Ch\_type&gt; Class Template Reference

## Public Types

- typedef `_RegexMask` **char\_class\_type**
- typedef `_Ch_type` **char\_type**
- typedef `std::locale` **locale\_type**
- typedef `std::basic_string< char_type >` **string\_type**

## Public Member Functions

- `regex_traits` ()
- `locale_type` `getloc` () const
- `locale_type` `imbue` (`locale_type` \_\_loc)
- bool `isctype` (`_Ch_type` \_\_c, `char_class_type` \_\_f) const
- template<typename `_Fwd_iter` >  
`char_class_type` `lookup_classname` (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last, bool \_\_icase=false) const
- template<typename `_Fwd_iter` >  
`string_type` `lookup_collatename` (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter` >  
`string_type` `transform` (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- template<typename `_Fwd_iter` >  
`string_type` `transform_primary` (`_Fwd_iter` \_\_first, `_Fwd_iter` \_\_last) const
- `char_type` `translate` (`char_type` \_\_c) const
- `char_type` `translate_nocase` (`char_type` \_\_c) const
- int `value` (`_Ch_type` \_\_ch, int \_\_radix) const

## Static Public Member Functions

- static `std::size_t` `length` (const `char_type` \*\_\_p)

## Protected Attributes

- `locale_type` **\_M\_locale**

## 4.889.1 Detailed Description

```
template<typename _Ch_type>
class std::regex_traits<_Ch_type>
```

Describes aspects of a regular expression.

A regular expression traits class that satisfies the requirements of section [28.7].

The class `regex` is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Definition at line 80 of file `regex.h`.

## 4.889.2 Constructor & Destructor Documentation

### 4.889.2.1 regex\_traits()

```
template<typename _Ch_type>
std::regex_traits< _Ch_type >::regex_traits () [inline]
```

Constructs a default traits object.

Definition at line 159 of file regex.h.

## 4.889.3 Member Function Documentation

### 4.889.3.1 getloc()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::getloc () const [inline]
```

Gets a copy of the current locale in use by the regex\_traits object.

Definition at line 372 of file regex.h.

### 4.889.3.2 imbue()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::imbue (
 locale_type __loc) [inline]
```

Imbues the regex\_traits object with a copy of a new locale.

#### Parameters

|                    |           |
|--------------------|-----------|
| <code>__loc</code> | A locale. |
|--------------------|-----------|

#### Returns

a copy of the previous locale in use by the regex\_traits object.



**Note**

Calling imbue with a different locale than the one currently in use invalidates all cached data held by \*this.

Definition at line 361 of file regex.h.

**4.889.3.3 isctype()**

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::isctype (
 _Ch_type __c,
 char_class_type __f) const
```

Determines if *c* is a member of an identified class.

**Parameters**

|                          |                                                   |
|--------------------------|---------------------------------------------------|
| $\leftrightarrow$<br>__c | a character.                                      |
| $\leftrightarrow$<br>__f | a class type (as returned from lookup_classname). |

**Returns**

true if the character \_\_c is a member of the classification represented by \_\_f, false otherwise.

**Exceptions**

|                      |                                                    |
|----------------------|----------------------------------------------------|
| <i>std::bad_cast</i> | if the current locale does not have a ctype facet. |
|----------------------|----------------------------------------------------|

**4.889.3.4 length()**

```
template<typename _Ch_type>
static std::size_t std::regex_traits< _Ch_type >::length (
 const char_type * __p) [inline], [static]
```

Gives the length of a C-style string starting at \_\_p.

**Parameters**

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| $\leftrightarrow$<br>__p | a pointer to the start of a character sequence. |
|--------------------------|-------------------------------------------------|

**Returns**

the number of characters between \*`__p` and the first default-initialized value of type `char_type`. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

Definition at line 172 of file `regex.h`.

**4.889.3.5 lookup\_classname()**

```
template<typename _Ch_type>
template<typename _Fwd_iter >
char_class_type std::regex_traits< _Ch_type >::lookup_classname (
 _Fwd_iter __first,
 _Fwd_iter __last,
 bool __icase = false) const
```

Maps one or more characters to a named character classification.

**Parameters**

|                      |                                              |
|----------------------|----------------------------------------------|
| <code>__first</code> | beginning of the character sequence.         |
| <code>__last</code>  | one-past-the-end of the character sequence.  |
| <code>__icase</code> | ignores the case of the classification name. |

**Returns**

an unspecified value that represents the character classification named by the character sequence designated by the iterator range `[__first, __last)`. If `icase` is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, `[[:lower:]]` is the same as `[[:alpha:]]`), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- `d`
- `w`
- `s`
- `alnum`
- `alpha`
- `blank`
- `cntrl`
- `digit`
- `graph`

- lower
- print
- punct
- space
- upper
- xdigit

#### 4.889.3.6 lookup\_collatename()

```
template<typename _Ch_type>
template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::lookup_collatename (
 _Fwd_iter __first,
 _Fwd_iter __last) const
```

Gets a collation element by name.

##### Parameters

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>__first</code> | beginning of the collation element name.        |
| <code>__last</code>  | one-past-the-end of the collation element name. |

##### Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [`__first`, `__last`). Returns an empty string if the character sequence is not a valid collating element.

#### 4.889.3.7 transform()

```
template<typename _Ch_type>
template<typename _Fwd_iter >
string_type std::regex_traits< _Ch_type >::transform (
 _Fwd_iter __first,
 _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence.

##### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then `v.transform(G1, G2) < v.transform(H1, H2)`.

What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

#### Returns

a locale-specific sort key equivalent to the input range.

#### Exceptions

|                            |                                                      |
|----------------------------|------------------------------------------------------|
| <code>std::bad_cast</code> | if the current locale does not have a collate facet. |
|----------------------------|------------------------------------------------------|

Definition at line 225 of file `regex.h`.

Referenced by `std::regex_traits<_CharType>::transform_primary()`.

#### 4.889.3.8 transform\_primary()

```
template<typename _Ch_type>
template<typename _Fwd_iter >
string_type std::regex_traits<_Ch_type>::transform_primary (
 _Fwd_iter __first,
 _Fwd_iter __last) const [inline]
```

Gets a sort key for a character sequence, independent of case.

#### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__first</code> | beginning of the character sequence.        |
| <code>__last</code>  | one-past-the-end of the character sequence. |

Effects: if `typeid(use_facet<collate<_Ch_type>>) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

**Todo** Implement this function correctly.

Definition at line 249 of file `regex.h`.

#### 4.889.3.9 translate()

```
template<typename _Ch_type>
char_type std::regex_traits<_Ch_type>::translate (
 char_type __c) const [inline]
```

Performs the identity translation.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

**Returns**

`__c`.

Definition at line 183 of file `regex.h`.

**4.889.3.10 `translate_nocase()`**

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate_nocase (
 char_type __c) const [inline]
```

Translates a character into a case-insensitive equivalent.

**Parameters**

|                  |                                                   |
|------------------|---------------------------------------------------|
| <code>__c</code> | A character to the locale-specific character set. |
|------------------|---------------------------------------------------|

**Returns**

the locale-specific lower-case equivalent of `__c`.

**Exceptions**

|                            |                                                        |
|----------------------------|--------------------------------------------------------|
| <code>std::bad_cast</code> | if the imbued locale does not support the ctype facet. |
|----------------------------|--------------------------------------------------------|

Definition at line 196 of file `regex.h`.

**4.889.3.11 `value()`**

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
 _Ch_type __ch,
 int __radix) const
```

Converts a digit to an int.

## Parameters

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| <code>__ch</code>    | a character representing a digit.                              |
| <code>__radix</code> | the radix if the numeric conversion (limited to 8, 10, or 16). |

## Returns

the value represented by the digit `__ch` in base `radix` if the character `__ch` is a valid digit in base `radix`; otherwise returns -1.

The documentation for this class was generated from the following file:

- [regex.h](#)

4.890 `std::remove_all_extents< _Tp >` Struct Template Reference

## Public Types

- `typedef _Tp type`

## 4.890.1 Detailed Description

```
template<typename _Tp>
struct std::remove_all_extents< _Tp >
```

`remove_all_extents`

Definition at line 785 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.891 `std::remove_const< _Tp >` Struct Template Reference

## Public Types

- `typedef _Tp type`

#### 4.891.1 Detailed Description

```
template<typename _Tp>
struct std::remove_const< _Tp >
```

remove\_const

Definition at line 1509 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.892 std::remove\_cv< \_Tp > Struct Template Reference

##### Public Types

- using **type** = \_Tp

#### 4.892.1 Detailed Description

```
template<typename _Tp>
struct std::remove_cv< _Tp >
```

remove\_cv

Definition at line 227 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.893 std::remove\_extent< \_Tp > Struct Template Reference

##### Public Types

- typedef \_Tp **type**



## 4.893.1 Detailed Description

```
template<typename _Tp>
struct std::remove_extent<_Tp>
```

`remove_extent`

Definition at line 1972 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.894 `std::remove_pointer<_Tp>` Struct Template Reference

Inherits `std::__remove_pointer_helper<_Tp, typename>`.

## Public Types

- `typedef _Tp type`

## 4.894.1 Detailed Description

```
template<typename _Tp>
struct std::remove_pointer<_Tp>
```

`remove_pointer`

Definition at line 2018 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.895 `std::remove_reference<_Tp>` Struct Template Reference

## Public Types

- `typedef _Tp type`

#### 4.895.1 Detailed Description

```
template<typename _Tp>
struct std::remove_reference< _Tp >
```

remove\_reference

Definition at line 1593 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

#### 4.896 std::remove\_volatile< \_Tp > Struct Template Reference

##### Public Types

- typedef \_Tp **type**

#### 4.896.1 Detailed Description

```
template<typename _Tp>
struct std::remove_volatile< _Tp >
```

remove\_volatile

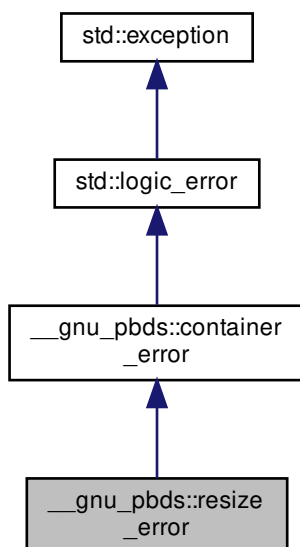
Definition at line 1518 of file type\_traits.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

## 4.897 \_\_gnu\_pbds::resize\_error Struct Reference

Inheritance diagram for \_\_gnu\_pbds::resize\_error:

**Public Member Functions**

- virtual const char \* [what](#) () const noexcept

## 4.897.1 Detailed Description

A container cannot be resized.

Definition at line 73 of file `exception.hpp`.

## 4.897.2 Member Function Documentation

#### 4.897.2.1 `what()`

```
virtual const char* std::logic_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::future\\_error](#).

The documentation for this struct was generated from the following file:

- [exception.hpp](#)

### 4.898 `__gnu_pbds::detail::resize_policy<_Tp>` Class Template Reference

#### Public Types

- typedef `_Tp` **size\_type**

#### Public Member Functions

- **resize\_policy** (const [resize\\_policy](#) &other)
- `size_type` **get\_new\_size\_for\_arbitrary** (`size_type`) const
- `size_type` **get\_new\_size\_for\_grow** () const
- `size_type` **get\_new\_size\_for\_shrink** () const
- `bool` **grow\_needed** (`size_type`) const
- `void` **notify\_arbitrary** (`size_type`)
- `void` **notify\_grow\_resize** ()
- `void` **notify\_shrink\_resize** ()
- `bool` **resize\_needed\_for\_grow** (`size_type`) const
- `bool` **resize\_needed\_for\_shrink** (`size_type`) const
- `bool` **shrink\_needed** (`size_type`) const
- `void` **swap** ([resize\\_policy](#)<`_Tp`> &)

#### Static Public Attributes

- static const `_Tp` **min\_size**

#### 4.898.1 Detailed Description

```
template<typename _Tp>
class __gnu_pbds::detail::resize_policy<_Tp>
```

Resize policy for binary heap.

Definition at line 52 of file `resize_policy.hpp`.

The documentation for this class was generated from the following file:

- [resize\\_policy.hpp](#)

## 4.899 std::result\_of&lt; \_Signature &gt; Class Template Reference

## 4.899.1 Detailed Description

```
template<typename _Signature>
class std::result_of< _Signature >
```

result\_of

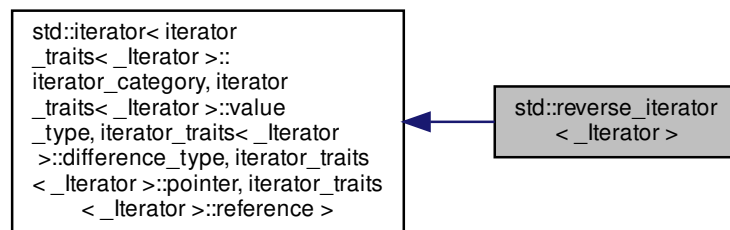
Definition at line 2344 of file type\_traits.

The documentation for this class was generated from the following file:

- [type\\_traits](#)

## 4.900 std::reverse\_iterator&lt; \_Iterator &gt; Class Template Reference

Inheritance diagram for std::reverse\_iterator< \_Iterator >:



## Public Types

- typedef \_\_traits\_type::difference\_type **difference\_type**
- typedef [iterator\\_traits](#)< \_Iterator >::iterator\_category **iterator\_category**
- typedef \_Iterator **iterator\_type**
- typedef \_\_traits\_type::pointer **pointer**
- typedef \_\_traits\_type::reference **reference**
- typedef [iterator\\_traits](#)< \_Iterator >::value\_type **value\_type**

## Public Member Functions

- constexpr [reverse\\_iterator](#) ()
- constexpr [reverse\\_iterator](#) (iterator\_type \_\_x)
- constexpr [reverse\\_iterator](#) (const [reverse\\_iterator](#) &\_\_x)
- template<typename \_Iter >  
constexpr [reverse\\_iterator](#) (const [reverse\\_iterator](#)< \_Iter > &\_\_x)
- constexpr iterator\_type [base](#) () const
- constexpr reference [operator\\*](#) () const
- constexpr [reverse\\_iterator](#) [operator+](#) (difference\_type \_\_n) const
- constexpr [reverse\\_iterator](#) & [operator++](#) ()
- constexpr [reverse\\_iterator](#) [operator++](#) (int)
- constexpr [reverse\\_iterator](#) & [operator+=](#) (difference\_type \_\_n)
- constexpr [reverse\\_iterator](#) [operator-](#) (difference\_type \_\_n) const
- constexpr [reverse\\_iterator](#) & [operator--](#) ()
- constexpr [reverse\\_iterator](#) [operator--](#) (int)
- constexpr [reverse\\_iterator](#) & [operator-=](#) (difference\_type \_\_n)
- constexpr pointer [operator->](#) () const
- [reverse\\_iterator](#) & [operator=](#) (const [reverse\\_iterator](#) &)=default
- constexpr reference [operator\[\]](#) (difference\_type \_\_n) const

## Protected Types

- typedef [iterator\\_traits](#)< \_Iterator > [\\_\\_traits\\_type](#)

## Protected Attributes

- [\\_Iterator](#) **current**

### 4.900.1 Detailed Description

```
template<typename _Iterator>
class std::reverse_iterator< _Iterator >
```

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

*This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2*

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

Definition at line 125 of file `bits/stl_iterator.h`.

## 4.900.2 Member Typedef Documentation

### 4.900.2.1 iterator\_category

```
typedef iterator_traits<_Iterator>::iterator_category std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::iterator_category [inherited]
```

One of the [tag types](#).

Definition at line 130 of file `stl_iterator_base_types.h`.

### 4.900.2.2 value\_type

```
typedef iterator_traits<_Iterator>::value_type std::iterator< iterator_traits<_Iterator>::iterator_category, iterator_traits<_Iterator>::value_type, iterator_traits<_Iterator>::difference_type, iterator_traits<_Iterator>::pointer, iterator_traits<_Iterator>::reference>::value_type [inherited]
```

The type "pointed to" by the iterator.

Definition at line 132 of file `stl_iterator_base_types.h`.

## 4.900.3 Constructor & Destructor Documentation

### 4.900.3.1 reverse\_iterator() [1/4]

```
template<typename _Iterator>
constexpr std::reverse_iterator<_Iterator>::reverse_iterator () [inline]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Definition at line 161 of file `bits/stl_iterator.h`.

Referenced by `std::reverse_iterator<_Iterator>::operator+()`, and `std::reverse_iterator<_Iterator>::operator-()`.

#### 4.900.3.2 reverse\_iterator() [2/4]

```
template<typename _Iterator>
constexpr std::reverse_iterator< _Iterator >::reverse_iterator (
 iterator_type __x) [inline], [explicit]
```

This iterator will move in the opposite direction that `x` does.

Definition at line 167 of file `bits/stl_iterator.h`.

#### 4.900.3.3 reverse\_iterator() [3/4]

```
template<typename _Iterator>
constexpr std::reverse_iterator< _Iterator >::reverse_iterator (
 const reverse_iterator< _Iterator > & __x) [inline]
```

The copy constructor is normal.

Definition at line 173 of file `bits/stl_iterator.h`.

#### 4.900.3.4 reverse\_iterator() [4/4]

```
template<typename _Iterator>
template<typename _Iter >
constexpr std::reverse_iterator< _Iterator >::reverse_iterator (
 const reverse_iterator< _Iter > & __x) [inline]
```

A `reverse_iterator` across other types can be copied if the underlying iterator can be converted to the type of `current`.

Definition at line 186 of file `bits/stl_iterator.h`.

### 4.900.4 Member Function Documentation

#### 4.900.4.1 base()

```
template<typename _Iterator>
constexpr iterator_type std::reverse_iterator< _Iterator >::base () const [inline]
```

##### Returns

`current`, the iterator used for underlying work.

Definition at line 193 of file `bits/stl_iterator.h`.

Referenced by `std::operator==()`.



#### 4.900.4.2 operator\*()

```
template<typename _Iterator>
constexpr reference std::reverse_iterator<_Iterator>::operator* () const [inline]
```

##### Returns

A reference to the value at `-current`

This requires that `-current` is dereferenceable.

##### Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: <http://gcc.gnu.org/PR51823>

Definition at line 207 of file `bits/stl_iterator.h`.

#### 4.900.4.3 operator+()

```
template<typename _Iterator>
constexpr reverse_iterator std::reverse_iterator<_Iterator>::operator+ (
 difference_type __n) const [inline]
```

##### Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 288 of file `bits/stl_iterator.h`.

References `std::reverse_iterator<_Iterator>::reverse_iterator()`.

#### 4.900.4.4 operator++() [1/2]

```
template<typename _Iterator>
constexpr reverse_iterator& std::reverse_iterator<_Iterator>::operator++ () [inline]
```

##### Returns

`*this`

Decrements the underlying iterator.

Definition at line 238 of file `bits/stl_iterator.h`.

#### 4.900.4.5 operator++() [2/2]

```
template<typename _Iterator>
constexpr reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
 int) [inline]
```

##### Returns

The original value of `*this`

Decrements the underlying iterator.

Definition at line 250 of file `bits/stl_iterator.h`.

#### 4.900.4.6 operator+=()

```
template<typename _Iterator>
constexpr reverse_iterator& std::reverse_iterator< _Iterator >::operator+= (
 difference_type __n) [inline]
```

##### Returns

`*this`

Moves the underlying iterator backwards `__n` steps. The underlying iterator must be a Random Access Iterator.

Definition at line 298 of file `bits/stl_iterator.h`.

#### 4.900.4.7 operator-()

```
template<typename _Iterator>
constexpr reverse_iterator std::reverse_iterator< _Iterator >::operator- (
 difference_type __n) const [inline]
```

##### Returns

A `reverse_iterator` that refers to `current - __n`

The underlying iterator must be a Random Access Iterator.

Definition at line 310 of file `bits/stl_iterator.h`.

References `std::reverse_iterator< _Iterator >::reverse_iterator()`.

## 4.900.4.8 operator--() [1/2]

```
template<typename _Iterator>
constexpr reverse_iterator& std::reverse_iterator<_Iterator>::operator-- () [inline]
```

**Returns**

\*this

Increments the underlying iterator.

Definition at line 263 of file bits/stl\_iterator.h.

## 4.900.4.9 operator--() [2/2]

```
template<typename _Iterator>
constexpr reverse_iterator& std::reverse_iterator<_Iterator>::operator-- (
 int) [inline]
```

**Returns**

A reverse\_iterator with the previous value of \*this

Increments the underlying iterator.

Definition at line 275 of file bits/stl\_iterator.h.

## 4.900.4.10 operator+=( )

```
template<typename _Iterator>
constexpr reverse_iterator& std::reverse_iterator<_Iterator>::operator+= (
 difference_type __n) [inline]
```

**Returns**

\*this

Moves the underlying iterator forwards \_\_n steps. The underlying iterator must be a Random Access Iterator.

Definition at line 320 of file bits/stl\_iterator.h.

4.900.4.11 `operator->()`

```
template<typename _Iterator>
constexpr pointer std::reverse_iterator< _Iterator >::operator-> () const [inline]
```

**Returns**

A pointer to the value at `-current`

This requires that `-current` is dereferenceable.

Definition at line 219 of file `bits/stl_iterator.h`.

4.900.4.12 `operator[]()`

```
template<typename _Iterator>
constexpr reference std::reverse_iterator< _Iterator >::operator[] (
 difference_type __n) const [inline]
```

**Returns**

The value at `current - __n - 1`

The underlying iterator must be a Random Access Iterator.

Definition at line 332 of file `bits/stl_iterator.h`.

The documentation for this class was generated from the following file:

- [bits/stl\\_iterator.h](#)

4.901 `__gnu_cxx::rope<_CharT, _Alloc>` Class Template Reference

Inherits `__gnu_cxx::Rope_base<_CharT, _Alloc>`.

**Public Types**

- `typedef _Rope_RopeConcatenation<_CharT, _Alloc> __C`
- `typedef _Rope_RopeFunction<_CharT, _Alloc> __F`
- `typedef _Rope_RopeLeaf<_CharT, _Alloc> __L`
- `typedef _Rope_RopeSubstring<_CharT, _Alloc> __S`
- `typedef \_\_alloc\_traits<_Alloc>::template rebind<__C>::other _CAlloc`
- `typedef \_\_alloc\_traits<_Alloc>::template rebind<_CharT>::other _DataAlloc`
- `typedef \_\_alloc\_traits<_Alloc>::template rebind<__F>::other _FAlloc`
- `typedef \_\_alloc\_traits<_Alloc>::template rebind<__L>::other _LAlloc`
- `typedef \_\_alloc\_traits<_Alloc>::template rebind<__S>::other _SAlloc`
- `typedef _Rope_const_iterator<_CharT, _Alloc> const_iterator`
- `typedef const _CharT * const_pointer`
- `typedef _CharT const_reference`
- `typedef std::reverse\_iterator<const_iterator> const_reverse_iterator`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Rope_iterator<_CharT, _Alloc> iterator`
- `typedef _Rope_char_ptr_proxy<_CharT, _Alloc> pointer`
- `typedef _Rope_char_ref_proxy<_CharT, _Alloc> reference`
- `typedef std::reverse\_iterator<iterator> reverse_iterator`
- `typedef std::size_t size_type`
- `typedef _CharT value_type`

## Public Member Functions

- **rope** (const `_CharT` \* \_\_s, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `_CharT` \* \_\_s, `size_type` \_\_len, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `_CharT` \* \_\_s, const `_CharT` \* \_\_e, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `const_iterator` & \_\_s, const `const_iterator` & \_\_e, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `iterator` & \_\_s, const `iterator` & \_\_e, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (`_CharT` \_\_c, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (`size_type` \_\_n, `_CharT` \_\_c, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `allocator_type` & \_\_a=allocator\_type())
- **rope** (`char_producer`< `_CharT` > \* \_\_fn, `size_type` \_\_len, bool \_\_delete\_fn, const `allocator_type` & \_\_a=allocator\_type())
- **rope** (const `rope` & \_\_x, const `allocator_type` & \_\_a=allocator\_type())
- `allocator_type` & **M\_get\_allocator** ()
- const `allocator_type` & **M\_get\_allocator** () const
- `rope` & **append** (const `_CharT` \* \_\_iter, `size_type` \_\_n)
- `rope` & **append** (const `_CharT` \* \_\_c\_string)
- `rope` & **append** (const `_CharT` \* \_\_s, const `_CharT` \* \_\_e)
- `rope` & **append** (const `iterator` \_\_s, const `iterator` \_\_e)
- `rope` & **append** (`_CharT` \_\_c)
- `rope` & **append** ()
- `rope` & **append** (const `rope` & \_\_y)
- `rope` & **append** (`size_type` \_\_n, `_CharT` \_\_c)
- void **apply\_to\_pieces** (`size_type` \_\_begin, `size_type` \_\_end, `_Rope_char_consumer`< `_CharT` > & \_\_c) const
- `_CharT` **at** (`size_type` \_\_pos) const
- `_CharT` **back** () const
- void **balance** ()
- const `iterator` **begin** () const
- const `iterator` **begin** ()
- const `_CharT` \* **c\_str** () const
- void **clear** ()
- int **compare** (const `rope` & \_\_y) const
- const `iterator` **const\_begin** () const
- const `iterator` **const\_end** () const
- const `reverse_iterator` **const\_rbegin** () const
- const `reverse_iterator` **const\_rend** () const
- void **copy** (`_CharT` \* \_\_buffer) const
- `size_type` **copy** (`size_type` \_\_pos, `size_type` \_\_n, `_CharT` \* \_\_buffer) const
- void **delete\_c\_str** ()
- void **dump** ()
- bool **empty** () const
- const `iterator` **end** () const
- const `iterator` **end** ()
- void **erase** (`size_type` \_\_p, `size_type` \_\_n)
- void **erase** (`size_type` \_\_p)
- `iterator` **erase** (const `iterator` & \_\_p, const `iterator` & \_\_q)
- `iterator` **erase** (const `iterator` & \_\_p)
- `size_type` **find** (`_CharT` \_\_c, `size_type` \_\_pos=0) const
- `size_type` **find** (const `_CharT` \* \_\_s, `size_type` \_\_pos=0) const
- `_CharT` **front** () const
- `allocator_type` **get\_allocator** () const

- void **insert** (size\_type \_\_p, const rope &\_\_r)
- void **insert** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_c\_string)
- void **insert** (size\_type \_\_p, \_CharT \_\_c)
- void **insert** (size\_type \_\_p)
- void **insert** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **insert** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **insert** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const rope &\_\_r)
- iterator **insert** (const iterator &\_\_p, size\_type \_\_n, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p, \_CharT \_\_c)
- iterator **insert** (const iterator &\_\_p)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- iterator **insert** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- iterator **insert** (const iterator &\_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- iterator **insert** (const iterator &\_\_p, const iterator &\_\_i, const iterator &\_\_j)
- size\_type **length** () const
- size\_type **max\_size** () const
- iterator **mutable\_begin** ()
- iterator **mutable\_end** ()
- **reverse\_iterator mutable\_rbegin** ()
- reference **mutable\_reference\_at** (size\_type \_\_pos)
- **reverse\_iterator mutable\_rend** ()
- rope & **operator=** (const rope &\_\_x)
- \_CharT **operator[]** (size\_type \_\_pos) const
- void **pop\_back** ()
- void **pop\_front** ()
- void **push\_back** (\_CharT \_\_x)
- void **push\_front** (\_CharT \_\_x)
- **const\_reverse\_iterator rbegin** () const
- **const\_reverse\_iterator rbegin** ()
- **const\_reverse\_iterator rend** () const
- **const\_reverse\_iterator rend** ()
- void **replace** (size\_type \_\_p, size\_type \_\_n, const rope &\_\_r)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, size\_type \_\_n, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, size\_type \_\_n, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (size\_type \_\_p, \_CharT \_\_c)
- void **replace** (size\_type \_\_p, const rope &\_\_r)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, size\_type \_\_i\_len)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (size\_type \_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (size\_type \_\_p, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (size\_type \_\_p, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const rope &\_\_r)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, \_CharT \_\_c)

- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const const\_iterator &\_\_i, const const\_iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const iterator &\_\_q, const iterator &\_\_i, const iterator &\_\_j)
- void **replace** (const iterator &\_\_p, const [rope](#) &\_\_r)
- void **replace** (const iterator &\_\_p, \_CharT \_\_c)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_c\_string)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, size\_type \_\_n)
- void **replace** (const iterator &\_\_p, const \_CharT \*\_\_i, const \_CharT \*\_\_j)
- void **replace** (const iterator &\_\_p, const\_iterator \_\_i, const\_iterator \_\_j)
- void **replace** (const iterator &\_\_p, iterator \_\_i, iterator \_\_j)
- const \_CharT \* **replace\_with\_c\_str** ()
- size\_type **size** () const
- [rope](#) **substr** (size\_type \_\_start, size\_type \_\_len=1) const
- [rope](#) **substr** (iterator \_\_start, iterator \_\_end) const
- [rope](#) **substr** (iterator \_\_start) const
- [rope](#) **substr** (const\_iterator \_\_start, const\_iterator \_\_end) const
- [rope](#)<\_CharT, \_Alloc> **substr** (const\_iterator \_\_start)
- void **swap** ([rope](#) &\_\_b)

#### Static Public Member Functions

- static \_\_C \* **\_C\_allocate** (std::size\_t \_\_n)
- static void **\_C\_deallocate** (\_\_C \*\_\_p, std::size\_t \_\_n)
- static \_CharT \* **\_Data\_allocate** (std::size\_t \_\_n)
- static void **\_Data\_deallocate** (\_CharT \*\_\_p, std::size\_t \_\_n)
- static \_\_F \* **\_F\_allocate** (std::size\_t \_\_n)
- static void **\_F\_deallocate** (\_\_F \*\_\_p, std::size\_t \_\_n)
- static \_\_L \* **\_L\_allocate** (std::size\_t \_\_n)
- static void **\_L\_deallocate** (\_\_L \*\_\_p, std::size\_t \_\_n)
- static \_\_S \* **\_S\_allocate** (std::size\_t \_\_n)
- static void **\_S\_deallocate** (\_\_S \*\_\_p, std::size\_t \_\_n)

#### Public Attributes

- \_RopeRep \* **\_M\_tree\_ptr**

#### Static Public Attributes

- static const size\_type **npos**

## Protected Types

- enum { **\_S\_copy\_max** }
- typedef \_Rope\_base< \_CharT, \_Alloc > **\_Base**
- typedef \_CharT \* **\_Cstrptr**
- typedef \_Rope\_RopeConcatenation< \_CharT, \_Alloc > **\_RopeConcatenation**
- typedef \_Rope\_RopeFunction< \_CharT, \_Alloc > **\_RopeFunction**
- typedef \_Rope\_RopeLeaf< \_CharT, \_Alloc > **\_RopeLeaf**
- typedef \_Rope\_RopeRep< \_CharT, \_Alloc > **\_RopeRep**
- typedef \_Rope\_RopeSubstring< \_CharT, \_Alloc > **\_RopeSubstring**
- typedef \_Rope\_self\_destruct\_ptr< \_CharT, \_Alloc > **\_Self\_destruct\_ptr**
- typedef \_Base::allocator\_type **allocator\_type**

## Static Protected Member Functions

- static size\_type **\_S\_allocated\_capacity** (size\_type \_\_n)
- static bool **\_S\_apply\_to\_pieces** (\_Rope\_char\_consumer< \_CharT > &\_\_c, const \_RopeRep \*\_\_r, size\_type \_\_begin, size\_type \_\_end)
- static \_RopeRep \* **\_S\_concat** (\_RopeRep \*\_\_left, \_RopeRep \*\_\_right)
- static \_RopeRep \* **\_S\_concat\_char\_iter** (\_RopeRep \*\_\_r, const \_CharT \*\_\_iter, size\_type \_\_slen)
- static \_RopeRep \* **\_S\_destr\_concat\_char\_iter** (\_RopeRep \*\_\_r, const \_CharT \*\_\_iter, size\_type \_\_slen)
- static \_RopeLeaf \* **\_S\_destr\_leaf\_concat\_char\_iter** (\_RopeLeaf \*\_\_r, const \_CharT \*\_\_iter, size\_type \_\_slen)
- static \_CharT **\_S\_fetch** (\_RopeRep \*\_\_r, size\_type \_\_pos)
- static \_CharT \* **\_S\_fetch\_ptr** (\_RopeRep \*\_\_r, size\_type \_\_pos)
- static bool **\_S\_is0** (\_CharT \_\_c)
- static \_RopeLeaf \* **\_S\_leaf\_concat\_char\_iter** (\_RopeLeaf \*\_\_r, const \_CharT \*\_\_iter, size\_type \_\_slen)
- static \_RopeConcatenation \* **\_S\_new\_RopeConcatenation** (\_RopeRep \*\_\_left, \_RopeRep \*\_\_right, allocator\_type &\_\_a)
- static \_RopeFunction \* **\_S\_new\_RopeFunction** (char\_producer< \_CharT > \*\_\_f, size\_type \_\_size, bool \_\_d, allocator\_type &\_\_a)
- static \_RopeLeaf \* **\_S\_new\_RopeLeaf** (\_CharT \*\_\_s, size\_type \_\_size, allocator\_type &\_\_a)
- static \_RopeSubstring \* **\_S\_new\_RopeSubstring** (\_Rope\_RopeRep< \_CharT, \_Alloc > \*\_\_b, size\_type \_\_s, size\_type \_\_l, allocator\_type &\_\_a)
- static void **\_S\_ref** (\_RopeRep \*\_\_t)
- static \_RopeLeaf \* **\_S\_RopeLeaf\_from\_unowned\_char\_ptr** (const \_CharT \*\_\_s, size\_type \_\_size, allocator\_type &\_\_a)
- static size\_type **\_S\_rounded\_up\_size** (size\_type \_\_n)
- static \_RopeRep \* **\_S\_substring** (\_RopeRep \*\_\_base, size\_type \_\_start, size\_type \_\_endp1)
- static \_RopeRep \* **\_S\_tree\_concat** (\_RopeRep \*\_\_left, \_RopeRep \*\_\_right)
- static void **\_S\_unref** (\_RopeRep \*\_\_t)
- static \_RopeRep \* **replace** (\_RopeRep \*\_\_old, size\_type \_\_pos1, size\_type \_\_pos2, \_RopeRep \*\_\_r)

## Static Protected Attributes

- static \_CharT **\_S\_empty\_c\_str** [1]



## Friends

- class `_Rope_char_ptr_proxy<_CharT, _Alloc>`
- class `_Rope_char_ref_proxy<_CharT, _Alloc>`
- class `_Rope_const_iterator<_CharT, _Alloc>`
- class `_Rope_iterator<_CharT, _Alloc>`
- class `_Rope_iterator_base<_CharT, _Alloc>`
- struct `_Rope_RopeRep<_CharT, _Alloc>`
- struct `_Rope_RopeSubstring<_CharT, _Alloc>`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const rope<_CharT2, _Alloc2> &__right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, const _CharT2 *__right)`
- template<class `_CharT2`, class `_Alloc2`>  
`rope<_CharT2, _Alloc2> operator+ (const rope<_CharT2, _Alloc2> &__left, _CharT2 __right)`

## 4.901.1 Detailed Description

```
template<class _CharT, class _Alloc>
class __gnu_cxx::rope<_CharT, _Alloc>
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html)

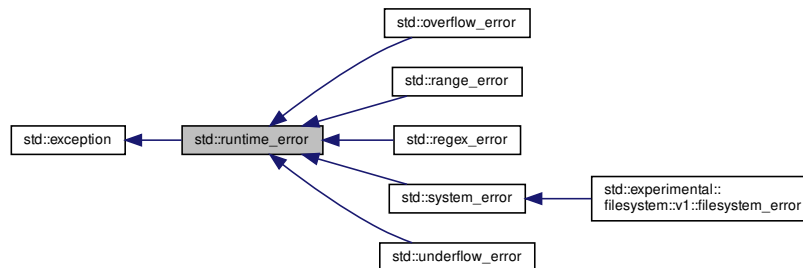
Definition at line 323 of file rope.

The documentation for this class was generated from the following files:

- [rope](#)
- [ropeimpl.h](#)

## 4.902 std::runtime\_error Class Reference

Inheritance diagram for `std::runtime_error`:



## Public Member Functions

- [runtime\\_error](#) (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- [runtime\\_error](#) (const char \*) \_GLIBCXX\_TXN\_SAFE
- [runtime\\_error](#) ([runtime\\_error](#) &&) noexcept
- [runtime\\_error](#) (const [runtime\\_error](#) &)=default
- [runtime\\_error](#) & [operator=](#) ([runtime\\_error](#) &&) noexcept
- [runtime\\_error](#) & [operator=](#) (const [runtime\\_error](#) &)=default
- virtual const char \* [what](#) () const noexcept

### 4.902.1 Detailed Description

One of two subclasses of exception.

Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

Definition at line 219 of file `stdexcept`.

### 4.902.2 Constructor & Destructor Documentation

#### 4.902.2.1 [runtime\\_error](#)()

```
std::runtime_error::runtime_error (
 const string & __arg) [explicit]
```

Takes a character string describing the error.

### 4.902.3 Member Function Documentation

#### 4.902.3.1 [what](#)()

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

## 4.903 \_\_gnu\_pbds::sample\_probe\_fn Class Reference

### Public Types

- typedef std::size\_t **size\_type**

### Public Member Functions

- [sample\\_probe\\_fn](#) ()
- [sample\\_probe\\_fn](#) (const [sample\\_probe\\_fn](#) &)
- void [swap](#) ([sample\\_probe\\_fn](#) &)

### Protected Member Functions

- size\_type [operator\(\)](#) (key\_const\_reference r\_key, size\_type i) const

#### 4.903.1 Detailed Description

A sample probe policy.

Definition at line 47 of file `sample_probe_fn.hpp`.

#### 4.903.2 Constructor & Destructor Documentation

##### 4.903.2.1 [sample\\_probe\\_fn\(\)](#) [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ()
```

Default constructor.

##### 4.903.2.2 [sample\\_probe\\_fn\(\)](#) [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (
 const sample_probe_fn &)
```

Copy constructor.

#### 4.903.3 Member Function Documentation

#### 4.903.3.1 operator()

```
size_type __gnu_pbds::sample_probe_fn::operator() (
 key_const_reference r_key,
 size_type i) const [inline], [protected]
```

Returns the i-th offset from the hash value of some key r\_key.

#### 4.903.3.2 swap()

```
void __gnu_pbds::sample_probe_fn::swap (
 sample_probe_fn &) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_probe\\_fn.hpp](#)

### 4.904 \_\_gnu\_pbds::sample\_range\_hashing Class Reference

#### Public Types

- typedef std::size\_t [size\\_type](#)

#### Public Member Functions

- [sample\\_range\\_hashing](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_range\\_hashing](#) &other)
- void [swap](#) ([sample\\_range\\_hashing](#) &other)

#### Protected Member Functions

- void [notify\\_resized](#) ([size\\_type](#))
- [size\\_type operator\(\)](#) ([size\\_type](#)) const

#### 4.904.1 Detailed Description

A sample range-hashing functor.

Definition at line 47 of file [sample\\_range\\_hashing.hpp](#).

## 4.904.2 Member Typedef Documentation

### 4.904.2.1 size\_type

```
typedef std::size_t __gnu_pbds::sample_range_hashing::size_type
```

Size type.

Definition at line 51 of file sample\_range\_hashing.hpp.

## 4.904.3 Constructor & Destructor Documentation

### 4.904.3.1 sample\_range\_hashing() [1/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing ()
```

Default constructor.

### 4.904.3.2 sample\_range\_hashing() [2/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing (
 const sample_range_hashing & other)
```

Copy constructor.

## 4.904.4 Member Function Documentation

### 4.904.4.1 notify\_resized()

```
void __gnu_pbds::sample_range_hashing::notify_resized (
 size_type) [protected]
```

Notifies the policy object that the container's size has changed to argument's size.

#### 4.904.4.2 operator()

```
size_type __gnu_pbds::sample_range_hashing::operator() (
 size_type) const [inline], [protected]
```

Transforms the \_\_hash value hash into a ranged-hash value.

#### 4.904.4.3 swap()

```
void __gnu_pbds::sample_range_hashing::swap (
 sample_range_hashing & other) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_range\\_hashing.hpp](#)

### 4.905 \_\_gnu\_pbds::sample\_ranged\_hash\_fn Class Reference

#### Public Types

- typedef std::size\_t **size\_type**

#### Public Member Functions

- [sample\\_ranged\\_hash\\_fn](#) ()
- [sample\\_ranged\\_hash\\_fn](#) (const [sample\\_ranged\\_hash\\_fn](#) &)
- void [swap](#) ([sample\\_ranged\\_hash\\_fn](#) &)

#### Protected Member Functions

- void [notify\\_resized](#) (size\_type)
- size\_type [operator\(\)](#) (key\_const\_reference) const

#### 4.905.1 Detailed Description

A sample ranged-hash functor.

Definition at line 47 of file [sample\\_ranged\\_hash\\_fn.hpp](#).

#### 4.905.2 Constructor & Destructor Documentation

#### 4.905.2.1 sample\_ranged\_hash\_fn() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()
```

Default constructor.

#### 4.905.2.2 sample\_ranged\_hash\_fn() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (
 const sample_ranged_hash_fn &)
```

Copy constructor.

### 4.905.3 Member Function Documentation

#### 4.905.3.1 notify\_resized()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (
 size_type) [protected]
```

Notifies the policy object that the container's \_\_size has changed to size.

#### 4.905.3.2 operator()()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (
 key_const_reference) const [inline], [protected]
```

Transforms key\_const\_reference into a position within the table.

#### 4.905.3.3 swap()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (
 sample_ranged_hash_fn &) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_hash\\_fn.hpp](#)

## 4.906 `__gnu_pbds::sample_ranged_probe_fn` Class Reference

### Public Types

- typedef std::size\_t **size\_type**

### Public Member Functions

- **sample\_ranged\_probe\_fn** (const [sample\\_ranged\\_probe\\_fn](#) &)
- void **swap** ([sample\\_ranged\\_probe\\_fn](#) &)

### Protected Member Functions

- void **notify\_resized** (size\_type)
- size\_type **operator()** (key\_const\_reference, std::size\_t, size\_type) const

#### 4.906.1 Detailed Description

A sample ranged-probe functor.

Definition at line 47 of file `sample_ranged_probe_fn.hpp`.

The documentation for this class was generated from the following file:

- [sample\\_ranged\\_probe\\_fn.hpp](#)

## 4.907 `__gnu_pbds::sample_resize_policy` Class Reference

### Public Types

- typedef std::size\_t [size\\_type](#)

### Public Member Functions

- [sample\\_resize\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_resize\\_policy](#) &other)
- void **swap** ([sample\\_resize\\_policy](#) &other)



### Protected Member Functions

- [size\\_type get\\_new\\_size](#) ([size\\_type](#) size, [size\\_type](#) num\_used\_e) const
- [bool is\\_resize\\_needed](#) () const
- [void notify\\_cleared](#) ()
- [void notify\\_erase\\_search\\_collision](#) ()
- [void notify\\_erase\\_search\\_end](#) ()
- [void notify\\_erase\\_search\\_start](#) ()
- [void notify\\_erased](#) ([size\\_type](#) num\_e)
- [void notify\\_find\\_search\\_collision](#) ()
- [void notify\\_find\\_search\\_end](#) ()
- [void notify\\_find\\_search\\_start](#) ()
- [void notify\\_insert\\_search\\_collision](#) ()
- [void notify\\_insert\\_search\\_end](#) ()
- [void notify\\_insert\\_search\\_start](#) ()
- [void notify\\_inserted](#) ([size\\_type](#) num\_e)
- [void notify\\_resized](#) ([size\\_type](#) new\_size)

#### 4.907.1 Detailed Description

A sample resize policy.

Definition at line 47 of file `sample_resize_policy.hpp`.

#### 4.907.2 Member Typedef Documentation

##### 4.907.2.1 size\_type

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```

Size type.

Definition at line 51 of file `sample_resize_policy.hpp`.

#### 4.907.3 Constructor & Destructor Documentation

##### 4.907.3.1 sample\_resize\_policy()

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ()
```

Default constructor.

#### 4.907.4 Member Function Documentation

##### 4.907.4.1 `get_new_size()`

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (
 size_type size,
 size_type num_used_e) const [protected]
```

Queries what the new size should be.

##### 4.907.4.2 `is_resize_needed()`

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

##### 4.907.4.3 `notify_cleared()`

```
void __gnu_pbds::sample_resize_policy::notify_cleared () [protected]
```

Notifies the table was cleared.

##### 4.907.4.4 `notify_erase_search_collision()`

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

##### 4.907.4.5 `notify_erase_search_end()`

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

#### 4.907.4.6 notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

#### 4.907.4.7 notify\_erased()

```
void __gnu_pbds::sample_resize_policy::notify_erased (
 size_type num_e) [inline], [protected]
```

Notifies an element was erased.

#### 4.907.4.8 notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### 4.907.4.9 notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

#### 4.907.4.10 notify\_find\_search\_start()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

#### 4.907.4.11 notify\_insert\_search\_collision()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**4.907.4.12 notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**4.907.4.13 notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**4.907.4.14 notify\_inserted()**

```
void __gnu_pbds::sample_resize_policy::notify_inserted (
 size_type num_e) [inline], [protected]
```

Notifies an element was inserted.

**4.907.4.15 notify\_resized()**

```
void __gnu_pbds::sample_resize_policy::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized to new\_size.

**4.907.4.16 sample\_range\_hashing()**

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (
 const sample_resize_policy & other)
```

Copy constructor.

**4.907.4.17 swap()**

```
void __gnu_pbds::sample_resize_policy::swap (
 sample_resize_policy & other) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_policy.hpp](#)

## 4.908 `__gnu_pbds::sample_resize_trigger` Class Reference

### Public Types

- typedef `std::size_t` `size_type`

### Public Member Functions

- `sample_resize_trigger` ()
- `sample_range_hashing` (const `sample_resize_trigger` &)
- void `swap` (`sample_resize_trigger` &)

### Protected Member Functions

- bool `is_grow_needed` (`size_type` size, `size_type` num\_entries) const
- bool `is_resize_needed` () const
- void `notify_cleared` ()
- void `notify_erase_search_collision` ()
- void `notify_erase_search_end` ()
- void `notify_erase_search_start` ()
- void `notify_erased` (`size_type` num\_entries)
- void `notify_externally_resized` (`size_type` new\_size)
- void `notify_find_search_collision` ()
- void `notify_find_search_end` ()
- void `notify_find_search_start` ()
- void `notify_insert_search_collision` ()
- void `notify_insert_search_end` ()
- void `notify_insert_search_start` ()
- void `notify_inserted` (`size_type` num\_entries)
- void `notify_resized` (`size_type` new\_size)

### 4.908.1 Detailed Description

A sample resize trigger policy.

Definition at line 47 of file `sample_resize_trigger.hpp`.

### 4.908.2 Member Typedef Documentation

#### 4.908.2.1 `size_type`

```
typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type
```

Size type.

Definition at line 51 of file `sample_resize_trigger.hpp`.

#### 4.908.3 Constructor & Destructor Documentation

##### 4.908.3.1 sample\_resize\_trigger()

```
__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()
```

Default constructor.

#### 4.908.4 Member Function Documentation

##### 4.908.4.1 is\_grow\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
 size_type size,
 size_type num_entries) const [inline], [protected]
```

Queries whether a grow is needed.

##### 4.908.4.2 is\_resize\_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const [inline], [protected]
```

Queries whether a resize is needed.

##### 4.908.4.3 notify\_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared () [protected]
```

Notifies the table was cleared.

##### 4.908.4.4 notify\_erase\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### 4.908.4.5 notify\_erase\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end () [inline], [protected]
```

Notifies a search ended.

#### 4.908.4.6 notify\_erase\_search\_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start () [inline], [protected]
```

Notifies a search started.

#### 4.908.4.7 notify\_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
 size_type num_entries) [inline], [protected]
```

Notifies an element was erased.

#### 4.908.4.8 notify\_externally\_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
 size_type new_size) [protected]
```

Notifies the table was resized externally.

#### 4.908.4.9 notify\_find\_search\_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

#### 4.908.4.10 notify\_find\_search\_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

**4.908.4.11 notify\_find\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start () [inline], [protected]
```

Notifies a search started.

**4.908.4.12 notify\_insert\_search\_collision()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

**4.908.4.13 notify\_insert\_search\_end()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

**4.908.4.14 notify\_insert\_search\_start()**

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start () [inline], [protected]
```

Notifies a search started.

**4.908.4.15 notify\_inserted()**

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
 size_type num_entries) [inline], [protected]
```

Notifies an element was inserted. the total number of entries in the table is num\_entries.

**4.908.4.16 notify\_resized()**

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
 size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.



## 4.908.4.17 sample\_range\_hashing()

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (
 const sample_resize_trigger &)
```

Copy constructor.

## 4.908.4.18 swap()

```
void __gnu_pbds::sample_resize_trigger::swap (
 sample_resize_trigger &) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_resize\\_trigger.hpp](#)

## 4.909 \_\_gnu\_pbds::sample\_size\_policy Class Reference

## Public Types

- typedef std::size\_t [size\\_type](#)

## Public Member Functions

- [sample\\_size\\_policy](#) ()
- [sample\\_range\\_hashing](#) (const [sample\\_size\\_policy](#) &)
- void [swap](#) ([sample\\_size\\_policy](#) &other)

## Protected Member Functions

- [size\\_type](#) [get\\_nearest\\_larger\\_size](#) ([size\\_type](#) size) const
- [size\\_type](#) [get\\_nearest\\_smaller\\_size](#) ([size\\_type](#) size) const

## 4.909.1 Detailed Description

A sample size policy.

Definition at line 47 of file [sample\\_size\\_policy.hpp](#).

## 4.909.2 Member Typedef Documentation

#### 4.909.2.1 size\_type

```
typedef std::size_t __gnu_pbds::sample_size_policy::size_type
```

Size type.

Definition at line 51 of file sample\_size\_policy.hpp.

### 4.909.3 Constructor & Destructor Documentation

#### 4.909.3.1 sample\_size\_policy()

```
__gnu_pbds::sample_size_policy::sample_size_policy ()
```

Default constructor.

### 4.909.4 Member Function Documentation

#### 4.909.4.1 get\_nearest\_larger\_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (
 size_type size) const [inline], [protected]
```

Given a \_\_size size, returns a \_\_size that is larger.

#### 4.909.4.2 get\_nearest\_smaller\_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (
 size_type size) const [inline], [protected]
```

Given a \_\_size size, returns a \_\_size that is smaller.

#### 4.909.4.3 sample\_range\_hashing()

```
__gnu_pbds::sample_size_policy::sample_range_hashing (
 const sample_size_policy &)
```

Copy constructor.

#### 4.909.4.4 `swap()`

```
void __gnu_pbds::sample_size_policy::swap (
 sample_size_policy & other) [inline]
```

Swaps content.

The documentation for this class was generated from the following file:

- [sample\\_size\\_policy.hpp](#)

### 4.910 `__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >` Class Template Reference

#### 4.910.1 Detailed Description

```
template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >
```

A sample node updator.

Definition at line 49 of file `sample_tree_node_update.hpp`.

The documentation for this class was generated from the following file:

- [sample\\_tree\\_node\\_update.hpp](#)

### 4.911 `__gnu_pbds::sample_trie_access_traits` Struct Reference

#### Public Types

- enum { **max\_size** }
- typedef `std::string::const_iterator` **const\_iterator**
- typedef char **e\_type**
- typedef `rebind_traits< _Alloc, key_type >::const_reference` **key\_const\_reference**
- typedef `std::string` **key\_type**
- typedef `std::size_t` **size\_type**

#### Static Public Member Functions

- static `const_iterator` **begin** (`key_const_reference`)
- static `size_type` **e\_pos** (`e_type`)
- static `const_iterator` **end** (`key_const_reference`)

#### 4.911.1 Detailed Description

A sample trie element access traits.

Definition at line 47 of file `sample_trie_access_traits.hpp`.

#### 4.911.2 Member Typedef Documentation

##### 4.911.2.1 `e_type`

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
```

Element type.

Definition at line 57 of file `sample_trie_access_traits.hpp`.

#### 4.911.3 Member Function Documentation

##### 4.911.3.1 `begin()`

```
static const_iterator __gnu_pbds::sample_trie_access_traits::begin (
 key_const_reference) [inline], [static]
```

Returns a `const_iterator` to the first element of `r_key`.

##### 4.911.3.2 `e_pos()`

```
static size_type __gnu_pbds::sample_trie_access_traits::e_pos (
 e_type) [inline], [static]
```

Maps an element to a position.

##### 4.911.3.3 `end()`

```
static const_iterator __gnu_pbds::sample_trie_access_traits::end (
 key_const_reference) [inline], [static]
```

Returns a `const_iterator` to the after-last element of `r_key`.

The documentation for this struct was generated from the following file:

- [sample\\_trie\\_access\\_traits.hpp](#)

## 4.912 `__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

### Public Types

- typedef std::size\_t **metadata\_type**

### Protected Member Functions

- [sample\\_trie\\_node\\_update](#) ()
- void [operator](#)() (node\_iterator, node\_const\_iterator) const

#### 4.912.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

A sample node updator.

Definition at line 49 of file `sample_trie_node_update.hpp`.

#### 4.912.2 Constructor & Destructor Documentation

##### 4.912.2.1 `sample_trie_node_update()`

```
template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_update
() [protected]
```

Default constructor.

#### 4.912.3 Member Function Documentation

##### 4.912.3.1 `operator()`

```
template<typename Node_Cltr , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::sample_trie_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >::operator() (
 node_iterator ,
 node_const_iterator) const [inline], [protected]
```

Updates the rank of a node through a node\_iterator `node_it`; `end_nd_it` is the end node iterator.

The documentation for this class was generated from the following file:

- [sample\\_trie\\_node\\_update.hpp](#)

## 4.913 \_\_gnu\_pbds::sample\_update\_policy Struct Reference

### Public Member Functions

- [sample\\_update\\_policy](#) ()
- [sample\\_update\\_policy](#) (const [sample\\_update\\_policy](#) &)
- void [swap](#) ([sample\\_update\\_policy](#) &other)

### Protected Types

- typedef some\_metadata\_type [metadata\\_type](#)

### Protected Member Functions

- [metadata\\_type operator\(\)](#) () const
- bool [operator\(\)](#) (metadata\_reference) const

#### 4.913.1 Detailed Description

A sample list-update policy.

Definition at line 47 of file [sample\\_update\\_policy.hpp](#).

#### 4.913.2 Member Typedef Documentation

##### 4.913.2.1 [metadata\\_type](#)

```
typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type [protected]
```

Metadata on which this functor operates.

Definition at line 61 of file [sample\\_update\\_policy.hpp](#).

#### 4.913.3 Constructor & Destructor Documentation

##### 4.913.3.1 [sample\\_update\\_policy\(\)](#) [1/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy ()
```

Default constructor.

#### 4.913.3.2 sample\_update\_policy() [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (
 const sample_update_policy &)
```

Copy constructor.

### 4.913.4 Member Function Documentation

#### 4.913.4.1 operator()() [1/2]

```
metadata_type __gnu_pbds::sample_update_policy::operator() () const [protected]
```

Creates a metadata object.

#### 4.913.4.2 operator()() [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (
 metadata_reference) const [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

#### 4.913.4.3 swap()

```
void __gnu_pbds::sample_update_policy::swap (
 sample_update_policy & other) [inline]
```

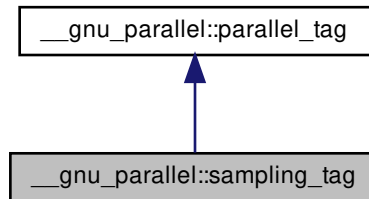
Swaps content.

The documentation for this struct was generated from the following file:

- [sample\\_update\\_policy.hpp](#)

#### 4.914 \_\_gnu\_parallel::sampling\_tag Struct Reference

Inheritance diagram for \_\_gnu\_parallel::sampling\_tag:



##### Public Member Functions

- **sampling\_tag** ([\\_ThreadIndex](#) \_\_num\_threads)
- [\\_ThreadIndex](#) **\_\_get\_num\_threads** ()
- void **set\_num\_threads** ([\\_ThreadIndex](#) \_\_num\_threads)

##### 4.914.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

Definition at line 118 of file tags.h.

##### 4.914.2 Member Function Documentation

###### 4.914.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

###### 4.914.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.



## Parameters

|                            |                            |
|----------------------------|----------------------------|
| <code>__num_threads</code> | Desired number of threads. |
|----------------------------|----------------------------|

Definition at line 73 of file `tags.h`.

The documentation for this struct was generated from the following file:

- [tags.h](#)

4.915 `std::scoped_allocator_adaptor<_OuterAlloc, _InnerAllocs>` Class Template Reference

Inherits `_OuterAlloc`.

## Public Types

- typedef `__traits::const_pointer` **const\_pointer**
- typedef `__traits::const_void_pointer` **const\_void\_pointer**
- typedef `__traits::difference_type` **difference\_type**
- typedef `__inner_type::__type` **inner\_allocator\_type**
- typedef `__and< typename __traits::is_always_equal, typename allocator_traits< _InnerAllocs >::is_always_equal... >::type` **is\_always\_equal**
- typedef `_OuterAlloc` **outer\_allocator\_type**
- typedef `__traits::pointer` **pointer**
- typedef `__or< typename __traits::propagate_on_container_copy_assignment, typename allocator_traits< _InnerAllocs >::propagate_on_container_copy_assignment... >::type` **propagate\_on\_container\_copy\_assignment**
- typedef `__or< typename __traits::propagate_on_container_move_assignment, typename allocator_traits< _InnerAllocs >::propagate_on_container_move_assignment... >::type` **propagate\_on\_container\_move\_assignment**
- typedef `__or< typename __traits::propagate_on_container_swap, typename allocator_traits< _InnerAllocs >::propagate_on_container_swap... >::type` **propagate\_on\_container\_swap**
- typedef `__traits::size_type` **size\_type**
- typedef `__traits::value_type` **value\_type**
- typedef `__traits::void_pointer` **void\_pointer**

## Public Member Functions

- template<typename `_Outer2`, typename `= _Constructible<_Outer2>>>`  
**scoped\_allocator\_adaptor** (`_Outer2` &&`__outer`, const `_InnerAllocs` &... `__inner`)
- **scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) &`__other`)
- **scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#) &&`__other`)
- template<typename `_Outer2`, typename `= _Constructible<const _Outer2>>>`  
**scoped\_allocator\_adaptor** (const [scoped\\_allocator\\_adaptor](#) < `_Outer2`, `_InnerAllocs`... > &`__other`)
- template<typename `_Outer2`, typename `= _Constructible<_Outer2>>>`  
**scoped\_allocator\_adaptor** ([scoped\\_allocator\\_adaptor](#) < `_Outer2`, `_InnerAllocs`... > &&`__other`)
- pointer **allocate** (`size_type` `__n`)

- pointer **allocate** (size\_type \_\_n, const\_void\_pointer \_\_hint)
- template<typename \_Tp, typename... \_Args>  
\_\_not\_pair< \_Tp >::type **construct** (\_Tp \*\_\_p, \_Args &&... \_\_args)
- template<typename \_T1, typename \_T2, typename... \_Args1, typename... \_Args2>  
void **construct** (pair< \_T1, \_T2 > \*\_\_p, [piecewise\\_construct\\_t](#), tuple< \_Args1... > \_\_x, tuple< \_Args2... > \_\_y)
- template<typename \_T1, typename \_T2 >  
void **construct** (pair< \_T1, \_T2 > \*\_\_p)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** (pair< \_T1, \_T2 > \*\_\_p, \_Up &&\_\_u, \_Vp &&\_\_v)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** (pair< \_T1, \_T2 > \*\_\_p, const pair< \_Up, \_Vp > &\_\_x)
- template<typename \_T1, typename \_T2, typename \_Up, typename \_Vp >  
void **construct** (pair< \_T1, \_T2 > \*\_\_p, pair< \_Up, \_Vp > &&\_\_x)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Tp >  
void **destroy** (\_Tp \*\_\_p)
- inner\_allocator\_type & **inner\_allocator** () noexcept
- const inner\_allocator\_type & **inner\_allocator** () const noexcept
- size\_type **max\_size** () const
- [scoped\\_allocator\\_adaptor](#) & **operator=** (const [scoped\\_allocator\\_adaptor](#) &)=default
- [scoped\\_allocator\\_adaptor](#) & **operator=** ([scoped\\_allocator\\_adaptor](#) &&)=default
- outer\_allocator\_type & **outer\_allocator** () noexcept
- const outer\_allocator\_type & **outer\_allocator** () const noexcept
- [scoped\\_allocator\\_adaptor](#) **select\_on\_container\_copy\_construction** () const

## Friends

- template<typename \_Outer, typename... \_Inner>  
class **scoped\_allocator\_adaptor**
- template<typename... >  
class **\_\_inner\_type\_impl**
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA... > &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA... > &\_\_b) noexcept

## Related Functions

(Note that these are not member functions.)

- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator!=** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA... > &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA... > &\_\_b) noexcept
- template<typename \_OutA1, typename \_OutA2, typename... \_InA>  
bool **operator==** (const [scoped\\_allocator\\_adaptor](#)< \_OutA1, \_InA... > &\_\_a, const [scoped\\_allocator\\_adaptor](#)< \_OutA2, \_InA... > &\_\_b) noexcept

## 4.915.1 Detailed Description

```
template<typename _OuterAlloc, typename... _InnerAllocs>
class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
```

An adaptor to recursively pass an allocator to the objects it constructs.

Definition at line 175 of file `scoped_allocator`.

The documentation for this class was generated from the following file:

- [scoped\\_allocator](#)

## 4.916 std::seed\_seq Class Reference

## Public Types

- typedef uint\_least32\_t [result\\_type](#)

## Public Member Functions

- [seed\\_seq](#) () noexcept
- template<typename \_IntType >  
**seed\_seq** (std::initializer\_list< \_IntType > \_\_il)
- template<typename \_InputIterator >  
**seed\_seq** (\_InputIterator \_\_begin, \_InputIterator \_\_end)
- **seed\_seq** (const [seed\\_seq](#) &)=delete
- template<typename \_RandomAccessIterator >  
void **generate** (\_RandomAccessIterator \_\_begin, \_RandomAccessIterator \_\_end)
- [seed\\_seq](#) & **operator=** (const [seed\\_seq](#) &)=delete
- template<typename \_OutputIterator >  
void **param** (\_OutputIterator \_\_dest) const
- size\_t **size** () const noexcept

## 4.916.1 Detailed Description

The `seed_seq` class generates sequences of seeds for random number generators.

Definition at line 6063 of file `random.h`.

## 4.916.2 Member Typedef Documentation

#### 4.916.2.1 result\_type

```
typedef uint_least32_t std::seed_seq::result_type
```

The type of the seed vales.

Definition at line 6067 of file random.h.

### 4.916.3 Constructor & Destructor Documentation

#### 4.916.3.1 seed\_seq()

```
std::seed_seq::seed_seq () [inline], [noexcept]
```

Default constructor.

Definition at line 6070 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.917 \_\_gnu\_cxx::select1st<\_Pair> Struct Template Reference

Inherits [std::\\_Select1st<\\_Pair>](#).

#### Public Types

- typedef [\\_Pair](#) [argument\\_type](#)
- typedef [\\_Pair::first\\_type](#) [result\\_type](#)

#### Public Member Functions

- [\\_Pair::first\\_type & operator\(\) \( \\_Pair &\\_\\_x\) const](#)
- [const \\_Pair::first\\_type & operator\(\) \(const \\_Pair &\\_\\_x\) const](#)
- [template<typename \\_Pair2 > \\_Pair2::first\\_type & operator\(\) \( \\_Pair2 &\\_\\_x\) const](#)
- [template<typename \\_Pair2 > const \\_Pair2::first\\_type & operator\(\) \(const \\_Pair2 &\\_\\_x\) const](#)

#### 4.917.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select1st< _Pair >
```

An [SGL extension](#) .

Definition at line 192 of file ext/functional.

#### 4.917.2 Member Typedef Documentation

##### 4.917.2.1 argument\_type

```
typedef _Pair std::unary_function< _Pair , _Pair::first_type >::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

##### 4.917.2.2 result\_type

```
typedef _Pair::first_type std::unary_function< _Pair , _Pair::first_type >::result_type [inherited]
```

result\_type is the return type

Definition at line 111 of file stl\_function.h.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

#### 4.918 \_\_gnu\_cxx::select2nd< \_Pair > Struct Template Reference

Inherits `std::_Select2nd< _Pair >`.

##### Public Types

- typedef \_Pair [argument\\_type](#)
- typedef \_Pair::second\_type [result\\_type](#)

## Public Member Functions

- `_Pair::second_type & operator() (_Pair &__x) const`
- `const _Pair::second_type & operator() (const _Pair &__x) const`

### 4.918.1 Detailed Description

```
template<class _Pair>
struct __gnu_cxx::select2nd< _Pair >
```

An [SGI extension](#) .

Definition at line 197 of file `ext/functional`.

### 4.918.2 Member Typedef Documentation

#### 4.918.2.1 `argument_type`

```
typedef _Pair std::unary_function< _Pair , _Pair::second_type >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.918.2.2 `result_type`

```
typedef _Pair::second_type std::unary_function< _Pair , _Pair::second_type >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [ext/functional](#)

## 4.919 `__gnu_pbds::detail::select_value_type< Key, Mapped >` Struct Template Reference

### Public Types

- typedef [std::pair](#)< const Key, Mapped > **type**

## 4.919.1 Detailed Description

```
template<typename Key, typename Mapped>
struct __gnu_pbds::detail::select_value_type< Key, Mapped >
```

Choose `value_type` to be a key/value pair or just a key.

Definition at line 107 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.920 `__gnu_pbds::detail::select_value_type< Key, null_type >` Struct Template Reference

## Public Types

- typedef Key **type**

## 4.920.1 Detailed Description

```
template<typename Key>
struct __gnu_pbds::detail::select_value_type< Key, null_type >
```

Specialization for sets where the key is the `value_type`.

Definition at line 114 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

4.921 `std::basic_istream< _CharT, _Traits >::sentry` Class Reference

## Public Types

- typedef `__istream_type::__ctype_type` **\_\_ctype\_type**
- typedef `_Traits::int_type` **\_\_int\_type**
- typedef `basic_istream< _CharT, _Traits >` **\_\_istream\_type**
- typedef `basic_streambuf< _CharT, _Traits >` **\_\_streambuf\_type**
- typedef `_Traits` **traits\_type**

## Public Member Functions

- [sentry](#) (`basic_istream< _CharT, _Traits > &__is`, `bool __noskipws=false`)
- [operator bool](#) () const

#### 4.921.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >::sentry
```

Performs setup work for input streams.

Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations*, although only prefix actions are currently required by the standard.

Definition at line 686 of file istream.

#### 4.921.2 Member Typedef Documentation

##### 4.921.2.1 traits\_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_istream< _CharT, _Traits >::sentry::traits_type
```

Easy access to dependent types.

Definition at line 693 of file istream.

#### 4.921.3 Constructor & Destructor Documentation

##### 4.921.3.1 sentry()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::sentry (
 basic_istream< _CharT, _Traits > & __is,
 bool __noskipws = false) [explicit]
```

The constructor performs all the work.

##### Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <code>__is</code>       | The input stream to guard.            |
| <code>__noskipws</code> | Whether to consume whitespace or not. |

If the stream state is good (`__is.good()` is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.



The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if `__noskipws` is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file `istream.tcc`.

References `std::basic_ios<_CharT, _Traits>::good()`, and `std::ios_base::goodbit`.

#### 4.921.4 Member Function Documentation

##### 4.921.4.1 `operator bool()`

```
template<typename _CharT, typename _Traits>
std::basic_istream<_CharT, _Traits>::sentry::operator bool () const [inline], [explicit]
```

Quick status checking.

##### Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (`true == okay`).

Definition at line 734 of file `istream`.

The documentation for this class was generated from the following files:

- [istream](#)
- [istream.tcc](#)

## 4.922 `std::basic_ostream<_CharT, _Traits>::sentry` Class Reference

### Public Member Functions

- [sentry](#) ([basic\\_ostream](#)<\_CharT, \_Traits> &\_\_os)
- [~sentry](#) ()
- [operator bool](#) () const

#### 4.922.1 Detailed Description

```
template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >::sentry
```

Performs setup work for output streams.

Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

Definition at line 432 of file ostream.

#### 4.922.2 Constructor & Destructor Documentation

##### 4.922.2.1 sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
 basic_ostream< _CharT, _Traits > & __os) [explicit]
```

The constructor performs preparatory work.

##### Parameters

|                   |                             |
|-------------------|-----------------------------|
| <code>__os</code> | The output stream to guard. |
|-------------------|-----------------------------|

If the stream state is good (`__os.good()` is true), then if the stream is tied to another output stream, `is-<tie()->flush()` is called to synchronize the output sequences.

If the stream state is still good, then the sentry state becomes true (*okay*).

Definition at line 47 of file ostream.tcc.

References `std::ios_base::failbit`, `std::basic_ios< _CharT, _Traits >::good()`, `std::basic_ios< _CharT, _Traits >::setstate()`, and `std::basic_ios< _CharT, _Traits >::tie()`.

##### 4.922.2.2 ~sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~sentry () [inline]
```

Possibly flushes the stream.

If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor calls `flush()` on the output stream.

Definition at line 462 of file ostream.

## 4.922.3 Member Function Documentation

## 4.922.3.1 operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::operator bool () const [inline], [explicit]
```

Quick status checking.

**Returns**

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

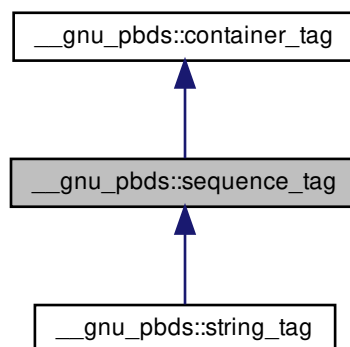
Definition at line 484 of file ostream.

The documentation for this class was generated from the following files:

- [ostream](#)
- [ostream.tcc](#)

## 4.923 \_\_gnu\_pbds::sequence\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::sequence\_tag:



#### 4.923.1 Detailed Description

Basic sequence.

Definition at line 129 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.924 \_\_gnu\_parallel::sequential\_tag Struct Reference

##### 4.924.1 Detailed Description

Forces sequential execution at compile time.

Definition at line 42 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

#### 4.925 std::set<\_Key, \_Compare, \_Alloc> Class Template Reference

##### Public Types

- typedef \_Key [key\\_type](#)
  - typedef \_Key [value\\_type](#)
  - typedef \_Compare [key\\_compare](#)
  - typedef \_Compare [value\\_compare](#)
  - typedef \_Alloc [allocator\\_type](#)
- 
- typedef \_Alloc\_traits::pointer [pointer](#)
  - typedef \_Alloc\_traits::const\_pointer [const\\_pointer](#)
  - typedef \_Alloc\_traits::reference [reference](#)
  - typedef \_Alloc\_traits::const\_reference [const\\_reference](#)
  - typedef \_Rep\_type::const\_iterator [iterator](#)
  - typedef \_Rep\_type::const\_iterator [const\\_iterator](#)
  - typedef \_Rep\_type::const\_reverse\_iterator [reverse\\_iterator](#)
  - typedef \_Rep\_type::const\_reverse\_iterator [const\\_reverse\\_iterator](#)
  - typedef \_Rep\_type::size\_type [size\\_type](#)
  - typedef \_Rep\_type::difference\_type [difference\\_type](#)

## Public Member Functions

- [set](#) ()=default
- [set](#) (const [\\_Compare](#) &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [template](#)<typename [\\_InputIterator](#) >  
[set](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [template](#)<typename [\\_InputIterator](#) >  
[set](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [\\_Compare](#) &\_\_comp, const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [set](#) &)=default
- [set](#) ([set](#) &&)=default
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [\\_Compare](#) &\_\_comp=[\\_Compare](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [set](#) (const [allocator\\_type](#) &\_\_a)
- [set](#) (const [set](#) &\_\_x, const [allocator\\_type](#) &\_\_a)
- [set](#) ([set](#) && \_\_x, const [allocator\\_type](#) &\_\_a) noexcept([is\\_nothrow\\_copy\\_constructible](#)< [\\_Compare](#) >::value &&\_\_a)↵  
[\\_Alloc\\_traits::S\\_always\\_equal](#)())
- [set](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, const [allocator\\_type](#) &\_\_a)
- [template](#)<typename [\\_InputIterator](#) >  
[set](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last, const [allocator\\_type](#) &\_\_a)
- [~set](#) ()=default
- [iterator begin](#) () const noexcept
- [iterator cbegin](#) () const noexcept
- [iterator cend](#) () const noexcept
- [void clear](#) () noexcept
- [reverse\\_iterator crbegin](#) () const noexcept
- [reverse\\_iterator crend](#) () const noexcept
- [template](#)<typename... [\\_Args](#) >  
[std::pair](#)< [iterator](#), bool > [emplace](#) ([\\_Args](#) &&... \_\_args)
- [template](#)<typename... [\\_Args](#) >  
[iterator emplace\\_hint](#) (const [iterator](#) \_\_pos, [\\_Args](#) &&... \_\_args)
- [bool empty](#) () const noexcept
- [iterator end](#) () const noexcept
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11 iterator erase](#) (const [iterator](#) \_\_position)
- [size\\_type erase](#) (const [key\\_type](#) &\_\_x)
- [\\_GLIBCXX\\_ABI\\_TAG\\_CXX11 iterator erase](#) (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
- [allocator\\_type get\\_allocator](#) () const noexcept
- [std::pair](#)< [iterator](#), bool > [insert](#) (const [value\\_type](#) &\_\_x)
- [std::pair](#)< [iterator](#), bool > [insert](#) ([value\\_type](#) &&\_\_x)
- [iterator insert](#) (const [iterator](#) \_\_position, const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const [iterator](#) \_\_position, [value\\_type](#) &&\_\_x)
- [template](#)<typename [\\_InputIterator](#) >  
[void insert](#) ([\\_InputIterator](#) \_\_first, [\\_InputIterator](#) \_\_last)
- [void insert](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [key\\_compare key\\_comp](#) () const
- [size\\_type max\\_size](#) () const noexcept
- [set & operator=](#) (const [set](#) &)=default
- [set & operator=](#) ([set](#) &&)=default
- [set & operator=](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
- [reverse\\_iterator rbegin](#) () const noexcept
- [reverse\\_iterator rend](#) () const noexcept

- `size_type size () const noexcept`
  - `void swap (set &__x) noexcept( /*conditional */)`
  - `value_compare value_comp () const`
- 
- `size_type count (const key_type &__x) const`
  - `template<typename _Kt >`  
`auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))`
- 
- `iterator find (const key_type &__x)`
  - `const_iterator find (const key_type &__x) const`
  - `template<typename _Kt >`  
`auto find (const _Kt &__x) -> decltype(iterator`
  - `template<typename _Kt >`  
`auto find (const _Kt &__x) const -> decltype(const_iterator`
- 
- `iterator lower_bound (const key_type &__x)`
  - `const_iterator lower_bound (const key_type &__x) const`
  - `template<typename _Kt >`  
`auto lower_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))`
  - `template<typename _Kt >`  
`auto lower_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))`
- 
- `iterator upper_bound (const key_type &__x)`
  - `const_iterator upper_bound (const key_type &__x) const`
  - `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))`
  - `template<typename _Kt >`  
`auto upper_bound (const _Kt &__x) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))`
- 
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
  - `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
  - `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))`
  - `template<typename _Kt >`  
`auto equal_range (const _Kt &__x) const -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))`

## Friends

- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`
- `template<typename _K1, typename _C1, typename _A1 >`  
`bool operator== (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)`

## 4.925.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::set< _Key, _Compare, _Alloc >
```

A standard container made up of unique keys, which can be retrieved in logarithmic time.

## Template Parameters

|                       |                                                                              |
|-----------------------|------------------------------------------------------------------------------|
| <code>_Key</code>     | Type of key objects.                                                         |
| <code>_Compare</code> | Comparison function object type, defaults to <code>less&lt;_Key&gt;</code> . |
| <code>_Alloc</code>   | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .             |

Meets the requirements of a [container](#), a [reversible container](#), and an [associative container](#) (using unique keys).

Sets support bidirectional iterators.

The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (\*\_unique versus \*\_equal, same as the standard).

Definition at line 70 of file `stl_multiset.h`.

## 4.925.2 Member Typedef Documentation

## 4.925.2.1 allocator\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typedef _Alloc std::set< _Key, _Compare, _Alloc >::allocator_type
```

Public typedefs.

Definition at line 124 of file `stl_set.h`.

#### 4.925.2.2 `const_iterator`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::const_iterator
```

Iterator-related typedefs.

Definition at line 148 of file `stl_set.h`.

#### 4.925.2.3 `const_pointer`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::const_pointer std::set< _Key, _Compare, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 141 of file `stl_set.h`.

#### 4.925.2.4 `const_reference`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::const_reference std::set< _Key, _Compare, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 143 of file `stl_set.h`.

#### 4.925.2.5 `const_reverse_iterator`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_iterator
```

Iterator-related typedefs.

Definition at line 150 of file `stl_set.h`.



#### 4.925.2.6 difference\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type
```

Iterator-related typedefs.

Definition at line 152 of file stl\_set.h.

#### 4.925.2.7 iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 147 of file stl\_set.h.

#### 4.925.2.8 key\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
```

Public typedefs.

Definition at line 122 of file stl\_set.h.

#### 4.925.2.9 key\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
```

Public typedefs.

Definition at line 109 of file stl\_set.h.

#### 4.925.2.10 pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 140 of file stl\_set.h.

#### 4.925.2.11 reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 142 of file stl\_set.h.

#### 4.925.2.12 reverse\_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
```

Iterator-related typedefs.

Definition at line 149 of file stl\_set.h.

#### 4.925.2.13 size\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 151 of file stl\_set.h.

## 4.925.2.14 value\_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typename _Compare std::set< _Key, _Compare, _Alloc >::value_compare
```

Public typedefs.

Definition at line 123 of file stl\_set.h.

## 4.925.2.15 value\_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
typename _Key std::set< _Key, _Compare, _Alloc >::value_type
```

Public typedefs.

Definition at line 121 of file stl\_set.h.

## 4.925.3 Constructor &amp; Destructor Documentation

## 4.925.3.1 set() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set () [default]
```

Default constructor creates no elements.

## 4.925.3.2 set() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a set with no elements.

Parameters

|                     |                      |
|---------------------|----------------------|
| <code>__comp</code> | Comparator to use.   |
| <code>__a</code>    | An allocator object. |

Definition at line 176 of file `stl_set.h`.

#### 4.925.3.3 `set()` [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set< _Key, _Compare, _Alloc >::set (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Builds a set from a range.

##### Parameters

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 191 of file `stl_set.h`.

#### 4.925.3.4 `set()` [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set< _Key, _Compare, _Alloc >::set (
 _InputIterator __first,
 _InputIterator __last,
 const _Compare & __comp,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from a range.

##### Parameters

|                      |                       |
|----------------------|-----------------------|
| <code>__first</code> | An input iterator.    |
| <code>__last</code>  | An input iterator.    |
| <code>__comp</code>  | A comparison functor. |
| <code>__a</code>     | An allocator object.  |

Create a set consisting of copies of the elements from `[__first,__last)`. This is linear in  $N$  if the range is already sorted, and  $N\log N$  otherwise (where  $N$  is `distance(__first,__last)`).

Definition at line 208 of file stl\_set.h.

#### 4.925.3.5 set() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 const set< _Key, _Compare, _Alloc > &) [default]
```

Set copy constructor.

Whether the allocator is copied depends on the allocator traits.

#### 4.925.3.6 set() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 set< _Key, _Compare, _Alloc > &&) [default]
```

Set move constructor

The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

#### 4.925.3.7 set() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 initializer_list< value_type > __l,
 const _Compare & __comp = _Compare(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a set from an initializer\_list.

##### Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__l</code>    | An initializer_list.  |
| <code>__comp</code> | A comparison functor. |
| <code>__a</code>    | An allocator object.  |

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is `__l.size()`).

Definition at line 243 of file stl\_set.h.

**4.925.3.8 set()** [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 const allocator_type & __a) [inline], [explicit]
```

Allocator-extended default constructor.

Definition at line 251 of file stl\_set.h.

**4.925.3.9 set()** [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 const set< _Key, _Compare, _Alloc > & __x,
 const allocator_type & __a) [inline]
```

Allocator-extended copy constructor.

Definition at line 255 of file stl\_set.h.

**4.925.3.10 set()** [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 set< _Key, _Compare, _Alloc > && __x,
 const allocator_type & __a) [inline], [noexcept]
```

Allocator-extended move constructor.

Definition at line 259 of file stl\_set.h.

**4.925.3.11 set()** [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::set (
 initializer_list< value_type > __l,
 const allocator_type & __a) [inline]
```

Allocator-extended initializer-list constructor.

Definition at line 265 of file stl\_set.h.

## 4.925.3.12 set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
std::set< _Key, _Compare, _Alloc >::set (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a) [inline]
```

Allocator-extended range constructor.

Definition at line 271 of file stl\_set.h.

## 4.925.3.13 ~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::set< _Key, _Compare, _Alloc >::~set () [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

## 4.925.4 Member Function Documentation

## 4.925.4.1 begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 344 of file stl\_set.h.

## 4.925.4.2 cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

Definition at line 381 of file stl\_set.h.

#### 4.925.4.3 cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 390 of file stl\_set.h.

#### 4.925.4.4 clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::set< _Key, _Compare, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 733 of file stl\_set.h.

#### 4.925.4.5 count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

##### Parameters

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

##### Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 748 of file stl\_set.h.



## 4.925.4.6 count() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::count (
 const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x)) [inline]
```

Finds the number of elements.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

## Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

Definition at line 754 of file stl\_set.h.

## 4.925.4.7 crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 399 of file stl\_set.h.

## 4.925.4.8 crend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 408 of file stl\_set.h.

## 4.925.4.9 emplace()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert an element into the set.

**Parameters**

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 462 of file `stl_set.h`.

**4.925.4.10 `emplace_hint()`**

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename... _Args>
iterator std::set<_Key, _Compare, _Alloc>::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to insert an element into the set.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 488 of file `stl_set.h`.

## 4.925.4.11 empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::set< _Key, _Compare, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the set is empty.

Definition at line 414 of file stl\_set.h.

## 4.925.4.12 end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

Definition at line 353 of file stl\_set.h.

## 4.925.4.13 equal\_range() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<iterator, iterator> std::set< _Key, _Compare, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

## Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

## Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 898 of file stl\_set.h.

**4.925.4.14** `equal_range()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair<const_iterator, const_iterator> std::set<_Key, _Compare, _Alloc >::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 902 of file `stl_set.h`.

**4.925.4.15** `equal_range()` [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set<_Key, _Compare, _Alloc >::equal_range (
 const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 908 of file stl\_set.h.

#### 4.925.4.16 equal\_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::equal_range (
 const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_
tr(__x))) [inline]
```

Finds a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
 c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multisets.

Definition at line 914 of file stl\_set.h.

#### 4.925.4.17 erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from a set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 654 of file `stl_set.h`.

**4.925.4.18 erase()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                                    |                              |
|------------------------------------|------------------------------|
| <code>↵</code><br><code>__x</code> | Key of element to be erased. |
|------------------------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 684 of file `stl_set.h`.

**4.925.4.19 erase()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from a set.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 706 of file `stl_set.h`.

## 4.925.4.20 find() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in a set.

## Parameters

|                  |                        |
|------------------|------------------------|
| <code>↵</code>   | Element to be located. |
| <code>__x</code> |                        |

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 794 of file `stl_set.h`.

## 4.925.4.21 find() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 798 of file `stl_set.h`.

**4.925.4.22 find()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::find (
 const _Kt & __x) -> decltype(iterator [inline]
```

Tries to locate an element in a set.

**Parameters**

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 804 of file `stl_set.h`.

**4.925.4.23 find()** [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↔
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::find (
 const _Kt & __x) const -> decltype(const_iterator [inline]
```

Tries to locate an element in a set.



## Parameters

|                 |                        |
|-----------------|------------------------|
| <code>_↵</code> | Element to be located. |
| <code>_X</code> |                        |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 810 of file `stl_set.h`.

## 4.925.4.24 get\_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator () const [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

Definition at line 335 of file `stl_set.h`.

## 4.925.4.25 insert() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
std::pair<iterator, bool> std::set< _Key, _Compare, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert an element into the set.

## Parameters

|                 |                         |
|-----------------|-------------------------|
| <code>_↵</code> | Element to be inserted. |
| <code>_X</code> |                         |

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.

Insertion requires logarithmic time.

Definition at line 509 of file `stl_set.h`.

References `std::pair<_T1, _T2>::first`, and `std::pair<_T1, _T2>::second`.

Referenced by `std::set<_Key, _Compare, _Alloc>::insert()`.

#### 4.925.4.26 `insert()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set<_Key, _Compare, _Alloc>::insert (
 const_iterator __position,
 const value_type & __x) [inline]
```

Attempts to insert an element into the set.

##### Parameters

|                         |                                                                               |
|-------------------------|-------------------------------------------------------------------------------|
| <code>__position</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>        | Element to be inserted.                                                       |

##### Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires logarithmic time (if the hint is not taken).

Definition at line 546 of file `stl_set.h`.

#### 4.925.4.27 `insert()` [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _InputIterator >
void std::set<_Key, _Compare, _Alloc>::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 566 of file stl\_set.h.

## 4.925.4.28 insert() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the set.

## Parameters

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| <code>↔</code> | A std::initializer_list<value_type> of elements to be inserted. |
| <code>↔</code> |                                                                 |
| <code>↔</code> |                                                                 |
| <code>↔</code> |                                                                 |
| <code>/</code> |                                                                 |

Complexity similar to that of the range constructor.

Definition at line 578 of file stl\_set.h.

References std::set< \_Key, \_Compare, \_Alloc >::insert().

## 4.925.4.29 key\_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::set< _Key, _Compare, _Alloc >::key_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

Definition at line 327 of file stl\_set.h.

## 4.925.4.30 lower\_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 829 of file `stl_set.h`.

**4.925.4.31 lower\_bound()** [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
 const key_type & __x) const [inline]
```

Finds the beginning of a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 833 of file `stl_set.h`.

**4.925.4.32 lower\_bound()** [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x))) [inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                   |                    |
|-------------------|--------------------|
| $\leftrightarrow$ | Key to be located. |
| $\_x$             |                    |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 839 of file stl\_set.h.

## 4.925.4.33 lower\_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<
_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
 const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

## Parameters

|                   |                    |
|-------------------|--------------------|
| $\leftrightarrow$ | Key to be located. |
| $\_x$             |                    |

## Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

Definition at line 845 of file stl\_set.h.

## 4.925.4.34 max\_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the set.

Definition at line 424 of file stl\_set.h.

**4.925.4.35 operator=()** [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
 const set< _Key, _Compare, _Alloc > &) [default]
```

Set assignment operator.

Whether the allocator is copied depends on the allocator traits.

**4.925.4.36 operator=()** [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
 set< _Key, _Compare, _Alloc > &&) [default]
```

Move assignment operator.

**4.925.4.37 operator=()** [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
set& std::set< _Key, _Compare, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Set list assignment operator.

**Parameters**

|    |                      |
|----|----------------------|
| ↵  | An initializer_list. |
| _↵ |                      |
| ↵  |                      |
| _↵ |                      |
| /  |                      |

This function fills a set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

Definition at line 316 of file stl\_set.h.

## 4.925.4.38 rbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

Definition at line 362 of file stl\_set.h.

## 4.925.4.39 rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

Definition at line 371 of file stl\_set.h.

## 4.925.4.40 size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the set.

Definition at line 419 of file stl\_set.h.

## 4.925.4.41 swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
void std::set< _Key, _Compare, _Alloc >::swap (
 set< _Key, _Compare, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another set.

## Parameters

|                  |                                                |
|------------------|------------------------------------------------|
| <code>__x</code> | A set of the same element and allocator types. |
|------------------|------------------------------------------------|

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

Definition at line 441 of file `stl_set.h`.

#### 4.925.4.42 `upper_bound()` [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or `end()`.

Definition at line 859 of file `stl_set.h`.

#### 4.925.4.43 `upper_bound()` [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↵
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
 const key_type & __x) const [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or `end()`.



Definition at line 863 of file stl\_set.h.

#### 4.925.4.44 upper\_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 869 of file stl\_set.h.

#### 4.925.4.45 upper\_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt >
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
 const _Kt & __x) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x))) [inline]
```

Finds the end of a subsequence matching given key.

##### Parameters

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

##### Returns

Iterator pointing to the first element greater than key, or end().

Definition at line 875 of file stl\_set.h.

4.925.4.46 `value_comp()`

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
value_compare std::set< _Key, _Compare, _Alloc >::value_comp () const [inline]
```

Returns the comparison object with which the set was constructed.

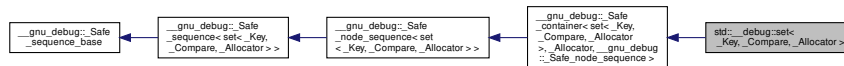
Definition at line 331 of file `stl_set.h`.

The documentation for this class was generated from the following files:

- [stl\\_multiset.h](#)
- [stl\\_set.h](#)

4.926 `std::__debug::set< _Key, _Compare, _Allocator >` Class Template Reference

Inheritance diagram for `std::__debug::set< _Key, _Compare, _Allocator >`:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_const_iterator, set >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug::__Safe_iterator< _Base_iterator, set >` **iterator**
- typedef `_Compare` **key\_compare**
- typedef `_Key` **key\_type**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Compare` **value\_compare**
- typedef `_Key` **value\_type**

## Public Member Functions

- `set` (const `set` &)=default
- `set` (`set` &&)=default
- `set` (`initializer_list`< `value_type` > \_\_l, const `_Compare` &\_\_comp=`_Compare`(), const `allocator_type` &\_\_a=`allocator_type`())
- `set` (const `allocator_type` &\_\_a)
- `set` (const `set` &\_\_x, const `allocator_type` &\_\_a)
- `set` (`set` &&\_\_x, const `allocator_type` &\_\_a) noexcept(noexcept(`_Base`(`std::move`(\_\_x.`_M_base`()), \_\_a)))
- `set` (`initializer_list`< `value_type` > \_\_l, const `allocator_type` &\_\_a)
- template<typename `_InputIterator` >  
`set` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `allocator_type` &\_\_a)
- `set` (const `_Compare` &\_\_comp, const `_Allocator` &\_\_a=`_Allocator`())
- template<typename `_InputIterator` >  
`set` (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const `_Compare` &\_\_comp=`_Compare`(), const `_Allocator` &\_\_a=`_Allocator`())
- `set` (const `_Base` &\_\_x)
- `_Base` & `_M_base` () noexcept
- const `_Base` & `_M_base` () const noexcept
- void `_M_invalidate_if` (`_Predicate` \_\_pred)
- void `_M_swap` (`_Safe_container` &\_\_x) noexcept
- void `_M_transfer_from_if` (`_Safe_sequence` &\_\_from, `_Predicate` \_\_pred)
- `iterator begin` () noexcept
- `const_iterator begin` () const noexcept
- `const_iterator cbegin` () const noexcept
- `const_iterator cend` () const noexcept
- void `clear` () noexcept
- `const_reverse_iterator crbegin` () const noexcept
- `const_reverse_iterator crend` () const noexcept
- template<typename... `_Args`>  
`std::pair`< `iterator`, bool > `emplace` (`_Args` &&... \_\_args)
- template<typename... `_Args`>  
`iterator emplace_hint` (const `iterator` \_\_pos, `_Args` &&... \_\_args)
- `iterator end` () noexcept
- `const_iterator end` () const noexcept
- `std::pair`< `iterator`, `iterator` > `equal_range` (const `key_type` &\_\_x)
- `std::pair`< `const_iterator`, `const_iterator` > `equal_range` (const `key_type` &\_\_x) const
- template<typename `_Kt`, typename `_Req` = typename `__has_is_transparent`<`_Compare`, `_Kt`>::type>  
`std::pair`< `iterator`, `iterator` > `equal_range` (const `_Kt` &\_\_x)
- template<typename `_Kt`, typename `_Req` = typename `__has_is_transparent`<`_Compare`, `_Kt`>::type>  
`std::pair`< `const_iterator`, `const_iterator` > `equal_range` (const `_Kt` &\_\_x) const
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const `iterator` \_\_position)
- `size_type erase` (const `key_type` &\_\_x)
- `_GLIBCXX_ABI_TAG_CXX11 iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `iterator find` (const `key_type` &\_\_x)
- `const_iterator find` (const `key_type` &\_\_x) const
- template<typename `_Kt`, typename `_Req` = typename `__has_is_transparent`<`_Compare`, `_Kt`>::type>  
`iterator find` (const `_Kt` &\_\_x)
- template<typename `_Kt`, typename `_Req` = typename `__has_is_transparent`<`_Compare`, `_Kt`>::type>  
`const_iterator find` (const `_Kt` &\_\_x) const
- `std::pair`< `iterator`, bool > `insert` (const `value_type` &\_\_x)

- `std::pair< iterator, bool > insert (value_type &&__x)`
- `iterator insert (const_iterator __position, const value_type &__x)`
- `iterator insert (const_iterator __position, value_type &&__x)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `void insert (initializer_list< value_type > __l)`
- `iterator lower_bound (const key_type &__x)`
- `const_iterator lower_bound (const key_type &__x) const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator lower_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator lower_bound (const _Kt &__x) const`
- `set & operator= (const set &)=default`
- `set & operator= (set &&)=default`
- `set & operator= (initializer_list< value_type > __l)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void swap (set &__x) noexcept(/*conditional */)`
- `iterator upper_bound (const key_type &__x)`
- `const_iterator upper_bound (const key_type &__x) const`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`iterator upper_bound (const _Kt &__x)`
- `template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>`  
`const_iterator upper_bound (const _Kt &__x) const`

#### Public Attributes

- `_Safe_iterator_base * _M_const_iterators`
- `_Safe_iterator_base * _M_iterators`
- `unsigned int _M_version`

#### Protected Member Functions

- `void _M_detach_all ()`
- `void _M_detach_singular ()`
- `__gnu_cxx::__mutex & _M_get_mutex () throw ()`
- `void _M_invalidate_all ()`
- `void _M_invalidate_all () const`
- `void _M_revalidate_singular ()`
- `_Safe_container & _M_safe () noexcept`
- `void _M_swap (_Safe_sequence_base &__x) noexcept`

#### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
`class ::__gnu_debug::_Safe_iterator`

#### 4.926.1 Detailed Description

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::set< _Key, _Compare, _Allocator >
```

Class std::set with safety/checking/debug instrumentation.

Definition at line 44 of file set.h.

#### 4.926.2 Member Function Documentation

##### 4.926.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by \_\_gnu\_debug::\_Safe\_sequence\_base::~~\_Safe\_sequence\_base().

##### 4.926.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

##### 4.926.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.926.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.926.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

#### 4.926.2.6 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.926.2.7 `_M_swap()`

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.926.2.8 `_M_transfer_from_if()`

```
void __gnu_debug::_Safe_sequence< set< _Key, _Compare, _Allocator > >::_M_transfer_from_if (
 _Safe_sequence< set< _Key, _Compare, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

### 4.926.3 Member Data Documentation

#### 4.926.3.1 \_M\_const\_iterators

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators` [inherited]

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.926.3.2 \_M\_iterators

`_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators` [inherited]

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.926.3.3 \_M\_version

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

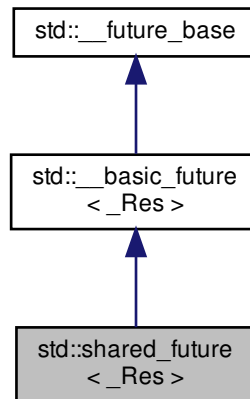
Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [set.h](#)

#### 4.927 `std::shared_future<_Res>` Class Template Reference

Inheritance diagram for `std::shared_future<_Res>`:



#### Public Types

- `template<typename _Res>`  
`using _Ptr = unique\_ptr<_Res, _Result_base::Deleter>`
- `using _State_base = _State_baseV2`

#### Public Member Functions

- `shared\_future (const shared\_future &__sf) noexcept`
- `shared\_future (future<_Res> &&__uf) noexcept`
- `shared\_future (shared\_future &&__sf) noexcept`
- `const _Res &get () const`
- `shared\_future & operator= (const shared\_future &__sf) noexcept`
- `shared\_future & operator= (shared\_future &&__sf) noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `template<typename _Rep, typename _Period>`  
`future\_status wait_for (const chrono::duration<_Rep, _Period> &__rel) const`
- `template<typename _Clock, typename _Duration>`  
`future\_status wait_until (const chrono::time\_point<_Clock, _Duration> &__abs) const`



## Static Public Member Functions

- template<typename \_Res, typename \_Allocator>  
static [\\_Ptr](#)<[\\_Result\\_alloc](#)<\_Res, \_Allocator>> [\\_S\\_allocate\\_result](#) (const \_Allocator &\_\_a)
- template<typename \_Res, typename \_Tp>  
static [\\_Ptr](#)<[\\_Result](#)<\_Res>> [\\_S\\_allocate\\_result](#) (const [std::allocator](#)<\_Tp> &\_\_a)
- template<typename \_BoundFn>  
static [std::shared\\_ptr](#)<\_State\_base> [\\_S\\_make\\_async\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_BoundFn>  
static [std::shared\\_ptr](#)<\_State\_base> [\\_S\\_make\\_deferred\\_state](#) (\_BoundFn &&\_\_fn)
- template<typename \_Res\_ptr, typename \_BoundFn>  
static [\\_Task\\_setter](#)<\_Res\_ptr, \_BoundFn> [\\_S\\_task\\_setter](#) (\_Res\_ptr &\_\_ptr, \_BoundFn &\_\_call)

## Protected Types

- typedef [\\_\\_future\\_base::Result](#)<\_Res> & [\\_\\_result\\_type](#)
- typedef [shared\\_ptr](#)<\_State\_base> [\\_\\_state\\_type](#)

## Protected Member Functions

- [\\_\\_result\\_type](#) [\\_M\\_get\\_result](#) () const
- void [\\_M\\_swap](#) ([\\_\\_basic\\_future](#) &\_\_that) noexcept

## 4.927.1 Detailed Description

```
template<typename _Res>
class std::shared_future<_Res>
```

Primary template for shared\_future.

Definition at line 128 of file future.

## 4.927.2 Member Typedef Documentation

4.927.2.1 [\\_Ptr](#)

```
template<typename _Res>
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A [unique\\_ptr](#) for result objects.

Definition at line 223 of file future.

### 4.927.3 Constructor & Destructor Documentation

#### 4.927.3.1 `shared_future()` [1/3]

```
template<typename _Res >
std::shared_future< _Res >::shared_future (
 const shared_future< _Res > & __sf) [inline], [noexcept]
```

Copy constructor.

Definition at line 910 of file future.

#### 4.927.3.2 `shared_future()` [2/3]

```
template<typename _Res >
std::shared_future< _Res >::shared_future (
 future< _Res > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 913 of file future.

#### 4.927.3.3 `shared_future()` [3/3]

```
template<typename _Res >
std::shared_future< _Res >::shared_future (
 shared_future< _Res > && __sf) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

Definition at line 918 of file future.

### 4.927.4 Member Function Documentation

#### 4.927.4.1 `_M_get_result()`

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file future.

## 4.927.4.2 get()

```
template<typename _Res >
const _Res& std::shared_future<_Res >::get () const [inline]
```

Retrieving the value.

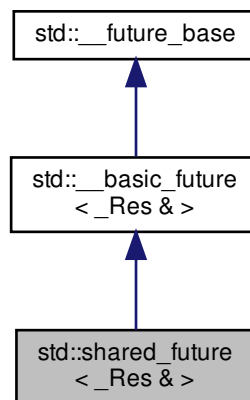
Definition at line 936 of file future.

The documentation for this class was generated from the following file:

- [future](#)

## 4.928 std::shared\_future&lt;\_Res &amp;&gt; Class Template Reference

Inheritance diagram for std::shared\_future<\_Res &>:



## Public Types

- `template<typename _Res >`  
  `using _Ptr = unique\_ptr<_Res, _Result_base::_Deleter >`
- `using _State_base = _State_baseV2`

## Public Member Functions

- `shared_future` (const `shared_future` &\_\_sf)
- `shared_future` (`future`< \_Res & > &&\_\_uf) noexcept
- `shared_future` (`shared_future` &&\_\_sf) noexcept
- `_Res` & `get` () const
- `shared_future` & `operator=` (const `shared_future` &\_\_sf)
- `shared_future` & `operator=` (`shared_future` &&\_\_sf) noexcept
- bool `valid` () const noexcept
- void `wait` () const
- `future_status` `wait_for` (const `chrono::duration`< \_Rep, \_Period > &\_\_rel) const
- `future_status` `wait_until` (const `chrono::time_point`< \_Clock, \_Duration > &\_\_abs) const

## Static Public Member Functions

- template<typename \_Res , typename \_Allocator >  
static `_Ptr`< `_Result_alloc`< \_Res, \_Allocator > > `_S_allocate_result` (const \_Allocator &\_\_a)
- template<typename \_Res , typename \_Tp >  
static `_Ptr`< `_Result`< \_Res > > `_S_allocate_result` (const `std::allocator`< \_Tp > &\_\_a)
- template<typename \_BoundFn >  
static `std::shared_ptr`< \_State\_base > `_S_make_async_state` (\_BoundFn &&\_\_fn)
- template<typename \_BoundFn >  
static `std::shared_ptr`< \_State\_base > `_S_make_deferred_state` (\_BoundFn &&\_\_fn)
- template<typename \_Res\_ptr , typename \_BoundFn >  
static `_Task_setter`< \_Res\_ptr, \_BoundFn > `_S_task_setter` (\_Res\_ptr &\_\_ptr, \_BoundFn &\_\_call)

## Protected Types

- typedef `__future_base::Result`< \_Res & > & `__result_type`
- typedef `shared_ptr`< \_State\_base > `__state_type`

## Protected Member Functions

- `__result_type` `_M_get_result` () const
- void `_M_swap` (`__basic_future` &\_\_that) noexcept

### 4.928.1 Detailed Description

```
template<typename _Res>
class std::shared_future< _Res & >
```

Partial specialization for `shared_future`<R&>

Definition at line 941 of file future.

## 4.928.2 Member Typedef Documentation

### 4.928.2.1 \_Ptr

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A unique\_ptr for result objects.

Definition at line 223 of file future.

## 4.928.3 Constructor & Destructor Documentation

### 4.928.3.1 shared\_future() [1/3]

```
template<typename _Res >
std::shared_future<_Res & >::shared_future (
 const shared_future<_Res & > & __sf) [inline]
```

Copy constructor.

Definition at line 949 of file future.

### 4.928.3.2 shared\_future() [2/3]

```
template<typename _Res >
std::shared_future<_Res & >::shared_future (
 future<_Res & > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 952 of file future.

### 4.928.3.3 shared\_future() [3/3]

```
template<typename _Res >
std::shared_future<_Res & >::shared_future (
 shared_future<_Res & > && __sf) [inline], [noexcept]
```

Construct from a shared\_future rvalue.

Definition at line 957 of file future.

#### 4.928.4 Member Function Documentation

##### 4.928.4.1 `_M_get_result()`

```
__result_type std::__basic_future< _Res & >::_M_get_result () const [inline], [protected],
[inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file future.

##### 4.928.4.2 `get()`

```
template<typename _Res >
_Res& std::shared_future< _Res & >::get () const [inline]
```

Retrieving the value.

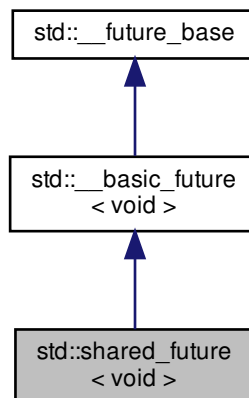
Definition at line 975 of file future.

The documentation for this class was generated from the following file:

- [future](#)

#### 4.929 `std::shared_future< void >` Class Template Reference

Inheritance diagram for `std::shared_future< void >`:



## Public Types

- `template<typename _Res >`  
`using _Ptr = unique\_ptr< _Res, Result\_base::Deleter >`
- `using \_State\_base = \_State\_baseV2`

## Public Member Functions

- `shared\_future (const shared\_future &__sf)`
- `shared\_future (future< void > &&__uf) noexcept`
- `shared\_future (shared\_future &&__sf) noexcept`
- `void get () const`
- `shared\_future & operator= (const shared\_future &__sf)`
- `shared\_future & operator= (shared\_future &&__sf) noexcept`
- `bool valid () const noexcept`
- `void wait () const`
- `future\_status wait\_for (const chrono::duration< _Rep, _Period > &__rel) const`
- `future\_status wait\_until (const chrono::time\_point< _Clock, _Duration > &__abs) const`

## Static Public Member Functions

- `template<typename _Res, typename _Allocator >`  
`static \_Ptr< \_Result\_alloc< _Res, _Allocator > > \_S\_allocate\_result (const _Allocator &__a)`
- `template<typename _Res, typename _Tp >`  
`static \_Ptr< \_Result< _Res > > \_S\_allocate\_result (const std::allocator< _Tp > &__a)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_async\_state (_BoundFn &&__fn)`
- `template<typename _BoundFn >`  
`static std::shared\_ptr< \_State\_base > \_S\_make\_deferred\_state (_BoundFn &&__fn)`
- `template<typename _Res_ptr, typename _BoundFn >`  
`static \_Task\_setter< _Res_ptr, _BoundFn > \_S\_task\_setter (_Res_ptr &__ptr, _BoundFn &__call)`

## Protected Types

- `typedef \_future\_base::Result< void > & \_\_result\_type`
- `typedef shared\_ptr< \_State\_base > \_\_state\_type`

## Protected Member Functions

- `\_\_result\_type \_M\_get\_result () const`
- `void \_M\_swap (\_\_basic\_future &__that) noexcept`

## 4.929.1 Detailed Description

```
template<>
class std::shared_future< void >
```

Explicit specialization for `shared_future<void>`

Definition at line 980 of file `future`.

#### 4.929.2 Member Typedef Documentation

##### 4.929.2.1 `_Ptr`

```
template<typename _Res >
using std::__future_base::_Ptr = unique_ptr<_Res, _Result_base::_Deleter> [inherited]
```

A `unique_ptr` for result objects.

Definition at line 223 of file `future`.

#### 4.929.3 Constructor & Destructor Documentation

##### 4.929.3.1 `shared_future()` [1/3]

```
std::shared_future< void >::shared_future (
 const shared_future< void > & __sf) [inline]
```

Copy constructor.

Definition at line 988 of file `future`.

##### 4.929.3.2 `shared_future()` [2/3]

```
std::shared_future< void >::shared_future (
 future< void > && __uf) [inline], [noexcept]
```

Construct from a future rvalue.

Definition at line 991 of file `future`.

##### 4.929.3.3 `shared_future()` [3/3]

```
std::shared_future< void >::shared_future (
 shared_future< void > && __sf) [inline], [noexcept]
```

Construct from a `shared_future` rvalue.

Definition at line 996 of file `future`.



## 4.929.4 Member Function Documentation

4.929.4.1 `_M_get_result()`

```
__result_type std::__basic_future< void >::_M_get_result () const [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

Definition at line 725 of file `future`.

The documentation for this class was generated from the following file:

- [future](#)

4.930 `std::shared_lock< _Mutex >` Class Template Reference

## Public Types

- typedef `_Mutex` **mutex\_type**

## Public Member Functions

- **shared\_lock** (`mutex_type &__m`)
- **shared\_lock** (`mutex_type &__m`, [defer\\_lock\\_t](#)) noexcept
- **shared\_lock** (`mutex_type &__m`, [try\\_to\\_lock\\_t](#))
- **shared\_lock** (`mutex_type &__m`, [adopt\\_lock\\_t](#))
- template<typename `_Clock`, typename `_Duration` >  
**shared\_lock** (`mutex_type &__m`, const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &`__abs_time`)
- template<typename `_Rep`, typename `_Period` >  
**shared\_lock** (`mutex_type &__m`, const [chrono::duration](#)< `_Rep`, `_Period` > &`__rel_time`)
- **shared\_lock** ([shared\\_lock](#) const &)=delete
- **shared\_lock** ([shared\\_lock](#) &&`__sl`) noexcept
- void **lock** ()
- `mutex_type * mutex` () const noexcept
- **operator bool** () const noexcept
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) const &)=delete
- [shared\\_lock](#) & **operator=** ([shared\\_lock](#) &&`__sl`) noexcept
- bool **owns\_lock** () const noexcept
- `mutex_type * release` () noexcept
- void **swap** ([shared\\_lock](#) &`__u`) noexcept
- bool **try\_lock** ()
- template<typename `_Rep`, typename `_Period` >  
bool **try\_lock\_for** (const [chrono::duration](#)< `_Rep`, `_Period` > &`__rel_time`)
- template<typename `_Clock`, typename `_Duration` >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< `_Clock`, `_Duration` > &`__abs_time`)
- void **unlock** ()

#### 4.930.1 Detailed Description

```
template<typename _Mutex>
class std::shared_lock< _Mutex >
```

shared\_lock

Definition at line 710 of file shared\_mutex.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

#### 4.931 std::shared\_ptr< \_Tp > Class Template Reference

Inherits std::\_\_shared\_ptr< \_Tp, \_Lp >.

##### Public Types

- using [element\\_type](#) = typename \_\_shared\_ptr< \_Tp >::[element\\_type](#)

##### Public Member Functions

- constexpr [shared\\_ptr](#) () noexcept
- [shared\\_ptr](#) (const [shared\\_ptr](#) &) noexcept=default
- template<typename \_Yp, typename = \_Constructible<\_Yp\*>>  
[shared\\_ptr](#) (\_Yp \*\_\_p)
- template<typename \_Yp, typename \_Deleter, typename = \_Constructible<\_Yp\*, \_Deleter>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d)
- template<typename \_Deleter >  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d)
- template<typename \_Yp, typename \_Deleter, typename \_Alloc, typename = \_Constructible<\_Yp\*, \_Deleter, \_Alloc>>  
[shared\\_ptr](#) (\_Yp \*\_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Deleter, typename \_Alloc >  
[shared\\_ptr](#) (nullptr\_t \_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- template<typename \_Yp >  
[shared\\_ptr](#) (const [shared\\_ptr](#)< \_Yp > &\_\_r, [element\\_type](#) \*\_\_p) noexcept
- template<typename \_Yp, typename = \_Constructible<const [shared\\_ptr](#)<\_Yp>&>>  
[shared\\_ptr](#) (const [shared\\_ptr](#)< \_Yp > &\_\_r) noexcept
- [shared\\_ptr](#) ([shared\\_ptr](#) &&\_\_r) noexcept
- template<typename \_Yp, typename = \_Constructible<[shared\\_ptr](#)<\_Yp>>>  
[shared\\_ptr](#) ([shared\\_ptr](#)< \_Yp > &&\_\_r) noexcept
- template<typename \_Tp1, typename >  
[shared\\_ptr](#) (std::auto\_ptr< \_Tp1 > &&\_\_r)
- template<typename \_Yp, typename = \_Constructible<const weak\_ptr<\_Yp>&>>  
[shared\\_ptr](#) (const [weak\\_ptr](#)< \_Yp > &\_\_r)

- template<typename \_Yp, typename \_Del, typename = \_Constructible<unique\_ptr<\_Yp, \_Del>>>>  
    **shared\_ptr** (**unique\_ptr**< \_Yp, \_Del > &&\_\_r)
- constexpr **shared\_ptr** (nullptr\_t) noexcept
- **element\_type** \* **get** () const noexcept
- **operator bool** () const
- **element\_type** & **operator\*** () const noexcept
- **element\_type** \* **operator->** () const noexcept
- **shared\_ptr** & **operator=** (const **shared\_ptr** &) noexcept=default
- template<typename \_Yp >  
    \_Assignable< const **shared\_ptr**< \_Yp > & > **operator=** (const **shared\_ptr**< \_Yp > &\_\_r) noexcept
- **shared\_ptr** & **operator=** (**shared\_ptr** &&\_\_r) noexcept
- template<class \_Yp >  
    \_Assignable< **shared\_ptr**< \_Yp > > **operator=** (**shared\_ptr**< \_Yp > &&\_\_r) noexcept
- template<typename \_Yp, typename \_Del >  
    \_Assignable< **unique\_ptr**< \_Yp, \_Del > > **operator=** (**unique\_ptr**< \_Yp, \_Del > &&\_\_r)
- void **reset** () noexcept
- template<typename \_Yp >  
    \_SafeConv< \_Yp > **reset** (\_Yp \* \_\_p)
- template<typename \_Yp, typename \_Deleter >  
    \_SafeConv< \_Yp > **reset** (\_Yp \* \_\_p, \_Deleter \_\_d)
- template<typename \_Yp, typename \_Deleter, typename \_Alloc >  
    \_SafeConv< \_Yp > **reset** (\_Yp \* \_\_p, \_Deleter \_\_d, \_Alloc \_\_a)
- void **swap** (\_\_shared\_ptr< \_Tp, \_Lp > &\_\_other) noexcept
- bool **unique** () const noexcept
- long **use\_count** () const noexcept
- template<typename \_Tp1 >  
    bool **owner\_before** (\_\_shared\_ptr< \_Tp1, \_Lp > const &\_\_rhs) const noexcept
- template<typename \_Tp1 >  
    bool **owner\_before** (\_\_weak\_ptr< \_Tp1, \_Lp > const &\_\_rhs) const noexcept

#### Friends

- template<typename \_Yp, typename \_Alloc, typename... \_Args>  
    **shared\_ptr**< \_Yp > **allocate\_shared** (const \_Alloc &\_\_a, \_Args &&... \_\_args)
- class **weak\_ptr**< \_Tp >

#### Related Functions

(Note that these are not member functions.)

- template<typename \_Del, typename \_Tp >  
    \_Del \* **get\_deleter** (const **shared\_ptr**< \_Tp > &\_\_p) noexcept
- template<typename \_Ch, typename \_Tr, typename \_Tp, \_Lock\_policy \_Lp>  
    std::basic\_ostream< \_Ch, \_Tr > & **operator<<** (std::basic\_ostream< \_Ch, \_Tr > &\_\_os, const \_\_shared\_ptr< \_Tp, \_Lp > &\_\_p)

- `template<typename _Tp, typename _Up >`  
`bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator!= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator< (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator<= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator> (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool operator>= (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept`
- `template<typename _Tp >`  
`bool operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp >`  
`void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp, typename _Up >`  
`shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up >`  
`shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
- `template<typename _Tp, typename _Up >`  
`shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept`
  
- `template<typename _Tp, typename _Alloc, typename... _Args>`  
`shared_ptr< _Tp > allocate_shared (const _Alloc &__a, _Args &&... __args)`

- template<typename \_Tp, typename... \_Args>  
shared\_ptr< \_Tp > make\_shared ( \_Args &&... \_\_args)
- template<typename \_Tp, \_Lock\_policy \_Lp>  
bool atomic\_is\_lock\_free (const \_\_shared\_ptr< \_Tp, \_Lp > \*\_\_p)
- template<typename \_Tp >  
shared\_ptr< \_Tp > atomic\_load\_explicit (const shared\_ptr< \_Tp > \*\_\_p, memory\_order)
- template<typename \_Tp >  
void atomic\_store\_explicit (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r, memory\_order)
- template<typename \_Tp >  
shared\_ptr< \_Tp > atomic\_exchange\_explicit (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \_\_r, memory\_order)
- template<typename \_Tp >  
bool atomic\_compare\_exchange\_strong\_explicit (shared\_ptr< \_Tp > \*\_\_p, shared\_ptr< \_Tp > \*\_\_v, shared\_ptr< \_Tp > \_\_w, memory\_order, memory\_order)

#### 4.931.1 Detailed Description

```
template<typename _Tp>
class std::shared_ptr< _Tp >
```

A smart pointer with reference-counted copy semantics.

A shared\_ptr object is either empty or *owns* a pointer passed to the constructor. Copies of a shared\_ptr share ownership of the same pointer. When the last shared\_ptr that owns the pointer is destroyed or reset, the owned pointer is freed (either by delete or by invoking a custom deleter that was passed to the constructor).

A shared\_ptr also stores another pointer, which is usually (but not always) the same pointer as it owns. The stored pointer can be retrieved by calling the get () member function.

The equality and relational operators for shared\_ptr only compare the stored pointer returned by get (), not the owned pointer. To test whether two shared\_ptr objects share ownership of the same pointer see std::shared\_ptr::owner\_before and std::owner\_less.

Definition at line 121 of file bits/shared\_ptr.h.

## 4.931.2 Member Typedef Documentation

### 4.931.2.1 element\_type

```
template<typename _Tp>
using std::shared_ptr< _Tp >::element_type = typename __shared_ptr<_Tp>::element_type
```

The type pointed to by the stored pointer, remove\_extent\_t<\_Tp>

Definition at line 136 of file bits/shared\_ptr.h.

## 4.931.3 Constructor & Destructor Documentation

### 4.931.3.1 shared\_ptr() [1/13]

```
template<typename _Tp>
constexpr std::shared_ptr< _Tp >::shared_ptr () [inline], [noexcept]
```

Construct an empty shared\_ptr.

#### Postcondition

use\_count()==0 && get()==0

Definition at line 147 of file bits/shared\_ptr.h.

### 4.931.3.2 shared\_ptr() [2/13]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
 const shared_ptr< _Tp > &) [default], [noexcept]
```

Copy constructor.

### 4.931.3.3 shared\_ptr() [3/13]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<_Yp*>>
std::shared_ptr< _Tp >::shared_ptr (
 _Yp * __p) [inline], [explicit]
```

Construct a shared\_ptr that owns the pointer \_\_p.

## Parameters

|                          |                                                 |
|--------------------------|-------------------------------------------------|
| $\leftrightarrow$<br>__p | A pointer that is convertible to element_type*. |
|--------------------------|-------------------------------------------------|

## Postcondition

use\_count() == 1 && get() == \_\_p

## Exceptions

|                    |                                  |
|--------------------|----------------------------------|
| std::bad_alloc, in | which case delete __p is called. |
|--------------------|----------------------------------|

Definition at line 159 of file bits/shared\_ptr.h.

## 4.931.3.4 shared\_ptr() [4/13]

```
template<typename _Tp>
template<typename _Yp , typename _Deleter , typename = _Constructible<_Yp*, _Deleter>>
std::shared_ptr< _Tp >::shared_ptr (
 _Yp * __p,
 _Deleter __d) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.

## Parameters

|                          |            |
|--------------------------|------------|
| $\leftrightarrow$<br>__p | A pointer. |
| $\leftrightarrow$<br>__d | A deleter. |

## Postcondition

use\_count() == 1 && get() == \_\_p

## Exceptions

|                    |                                |
|--------------------|--------------------------------|
| std::bad_alloc, in | which case __d(__p) is called. |
|--------------------|--------------------------------|

Requirements: \_Deleter's copy constructor and destructor must not throw

\_\_shared\_ptr will release \_\_p by calling \_\_d(\_\_p)

Definition at line 176 of file bits/shared\_ptr.h.

#### 4.931.3.5 shared\_ptr() [5/13]

```
template<typename _Tp>
template<typename _Deleter >
std::shared_ptr< _Tp >::shared_ptr (
 nullptr_t __p,
 _Deleter __d) [inline]
```

Construct a shared\_ptr that owns a null pointer and the deleter \_\_d.

##### Parameters

|                          |                          |
|--------------------------|--------------------------|
| $\leftrightarrow$<br>__p | A null pointer constant. |
| $\leftrightarrow$<br>__d | A deleter.               |

##### Postcondition

use\_count() == 1 && get() == \_\_p

##### Exceptions

|                            |                                |
|----------------------------|--------------------------------|
| <i>std::bad_alloc</i> , in | which case __d(__p) is called. |
|----------------------------|--------------------------------|

Requirements: \_Deleter's copy constructor and destructor must not throw

The last owner will call \_\_d(\_\_p)

Definition at line 193 of file bits/shared\_ptr.h.

#### 4.931.3.6 shared\_ptr() [6/13]

```
template<typename _Tp>
template<typename _Yp , typename _Deleter , typename _Alloc , typename = _Constructible<_Yp*, \leftrightarrow
_Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
 _Yp * __p,
 _Deleter __d,
 _Alloc __a) [inline]
```

Construct a shared\_ptr that owns the pointer \_\_p and the deleter \_\_d.



## Parameters

|                  |               |
|------------------|---------------|
| <code>__p</code> | A pointer.    |
| <code>__d</code> | A deleter.    |
| <code>__a</code> | An allocator. |

## Postcondition

`use_count() == 1 && get() == __p`

## Exceptions

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

`__shared_ptr` will release `__p` by calling `__d(__p)`

Definition at line 213 of file `bits/shared_ptr.h`.

4.931.3.7 `shared_ptr()` [7/13]

```
template<typename _Tp>
template<typename _Deleter , typename _Alloc >
std::shared_ptr<_Tp>::shared_ptr (
 nullptr_t __p,
 _Deleter __d,
 _Alloc __a) [inline]
```

Construct a `shared_ptr` that owns a null pointer and the deleter `__d`.

## Parameters

|                  |                          |
|------------------|--------------------------|
| <code>__p</code> | A null pointer constant. |
| <code>__d</code> | A deleter.               |
| <code>__a</code> | An allocator.            |

**Postcondition**

```
use_count() == 1 && get() == __p
```

**Exceptions**

|                                  |                                             |
|----------------------------------|---------------------------------------------|
| <code>std::bad_alloc</code> , in | which case <code>__d(__p)</code> is called. |
|----------------------------------|---------------------------------------------|

Requirements: `_Deleter`'s copy constructor and destructor must not throw `_Alloc`'s copy constructor and destructor must not throw.

The last owner will call `__d(__p)`

Definition at line 232 of file `bits/shared_ptr.h`.

**4.931.3.8 shared\_ptr()** [8/13]

```
template<typename _Tp>
template<typename _Yp >
std::shared_ptr< _Tp >::shared_ptr (
 const shared_ptr< _Yp > & __r,
 element_type * __p) [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

**Parameters**

|                  |                                                                    |
|------------------|--------------------------------------------------------------------|
| <code>__r</code> | A <code>shared_ptr</code> .                                        |
| <code>__p</code> | A pointer that will remain valid while <code>*__r</code> is valid. |

**Postcondition**

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

Definition at line 256 of file `bits/shared_ptr.h`.

## 4.931.3.9 shared\_ptr() [9/13]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr<_Tp>::shared_ptr (
 const shared_ptr<_Yp> & __r) [inline], [noexcept]
```

If `__r` is empty, constructs an empty `shared_ptr`; otherwise construct a `shared_ptr` that shares ownership with `__r`.

## Parameters

|          |               |
|----------|---------------|
| ↩        | A shared_ptr. |
| __↩      |               |
| ↩        |               |
| __↩      |               |
| <i>r</i> |               |

## Postcondition

`get() == __r.get() && use_count() == __r.use_count()`

Definition at line 295 of file `bits/shared_ptr.h`.

## 4.931.3.10 shared\_ptr() [10/13]

```
template<typename _Tp>
std::shared_ptr<_Tp>::shared_ptr (
 shared_ptr<_Tp> && __r) [inline], [noexcept]
```

Move-constructs a `shared_ptr` instance from `__r`.

## Parameters

|          |                      |
|----------|----------------------|
| ↩        | A shared_ptr rvalue. |
| __↩      |                      |
| ↩        |                      |
| __↩      |                      |
| <i>r</i> |                      |

## Postcondition

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 303 of file `bits/shared_ptr.h`.

**4.931.3.11** `shared_ptr()` [11/13]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr< _Tp >::shared_ptr (
 shared_ptr< _Yp > && __r) [inline], [noexcept]
```

Move-constructs a `shared_ptr` instance from `__r`.

**Parameters**

|          |                                   |
|----------|-----------------------------------|
| ↩        | A <code>shared_ptr</code> rvalue. |
| ↩        |                                   |
| ↩        |                                   |
| ↩        |                                   |
| <i>r</i> |                                   |

**Postcondition**

\*this contains the old value of `__r`, `__r` is empty.

Definition at line 312 of file `bits/shared_ptr.h`.

**4.931.3.12** `shared_ptr()` [12/13]

```
template<typename _Tp>
template<typename _Yp , typename = _Constructible<const weak_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
 const weak_ptr< _Yp > & __r) [inline], [explicit]
```

Constructs a `shared_ptr` that shares ownership with `__r` and stores a copy of the pointer stored in `__r`.

**Parameters**

|          |                           |
|----------|---------------------------|
| ↩        | A <code>weak_ptr</code> . |
| ↩        |                           |
| ↩        |                           |
| ↩        |                           |
| <i>r</i> |                           |

**Postcondition**

`use_count() == __r.use_count()`

**Exceptions**

|                     |                                                                                |
|---------------------|--------------------------------------------------------------------------------|
| <i>bad_weak_ptr</i> | when <code>__r.expired()</code> , in which case the constructor has no effect. |
|---------------------|--------------------------------------------------------------------------------|

Definition at line 324 of file bits/shared\_ptr.h.

#### 4.931.3.13 shared\_ptr() [13/13]

```
template<typename _Tp>
constexpr std::shared_ptr< _Tp >::shared_ptr (
 nullptr_t) [inline], [noexcept]
```

Construct an empty shared\_ptr.

##### Postcondition

use\_count() == 0 && get() == nullptr

Definition at line 356 of file bits/shared\_ptr.h.

#### 4.931.4 Member Function Documentation

##### 4.931.4.1 get()

```
template<typename _Tp, _Lock_policy _Lp>
element_type* std::__shared_ptr< _Tp, _Lp >::get (
 void) const [inline], [noexcept], [inherited]
```

Return the stored pointer.

Definition at line 1324 of file shared\_ptr\_base.h.

##### 4.931.4.2 operator bool()

```
template<typename _Tp, _Lock_policy _Lp>
std::__shared_ptr< _Tp, _Lp >::operator bool () const [inline], [explicit], [inherited]
```

Return true if the stored pointer is not null.

Definition at line 1328 of file shared\_ptr\_base.h.

**4.931.4.3 owner\_before()** [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1 >
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
 __shared_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

Definition at line 1358 of file `shared_ptr_base.h`.

**4.931.4.4 owner\_before()** [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1 >
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
 __weak_ptr< _Tp1, _Lp > const & __rhs) const [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two `shared_ptr` or `weak_ptr` objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

Definition at line 1363 of file `shared_ptr_base.h`.

**4.931.4.5 swap()**

```
template<typename _Tp, _Lock_policy _Lp>
void std::__shared_ptr< _Tp, _Lp >::swap (
 __shared_ptr< _Tp, _Lp > & __other) [inline], [noexcept], [inherited]
```

Exchange both the owned pointer and the stored pointer.

Definition at line 1343 of file `shared_ptr_base.h`.

**4.931.4.6 unique()**

```
template<typename _Tp, _Lock_policy _Lp>
bool std::__shared_ptr< _Tp, _Lp >::unique () const [inline], [noexcept], [inherited]
```

Return true if `use_count() == 1`.

Definition at line 1333 of file `shared_ptr_base.h`.

## 4.931.4.7 use\_count()

```
template<typename _Tp, _Lock_policy _Lp>
long std::__shared_ptr< _Tp, _Lp >::use_count () const [inline], [noexcept], [inherited]
```

If \*this owns a pointer, return the number of owners, otherwise zero.

Definition at line 1338 of file shared\_ptr\_base.h.

The documentation for this class was generated from the following files:

- [bits/shared\\_ptr.h](#)
- [shared\\_ptr\\_atomic.h](#)
- [auto\\_ptr.h](#)

## 4.932 std::shared\_timed\_mutex Class Reference

Inherits `__shared_timed_mutex_base`.

## Public Member Functions

- **shared\_timed\_mutex** (const [shared\\_timed\\_mutex](#) &)=delete
- void **lock** ()
- void **lock\_shared** ()
- [shared\\_timed\\_mutex](#) & **operator=** (const [shared\\_timed\\_mutex](#) &)=delete
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- bool **try\_lock\_shared** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_shared\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_shared\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_abs\_time)
- void **unlock** ()
- void **unlock\_shared** ()

## 4.932.1 Detailed Description

The standard shared timed mutex type.

Definition at line 447 of file shared\_mutex.

The documentation for this class was generated from the following file:

- [shared\\_mutex](#)

## 4.933 `std::shuffle_order_engine<_RandomNumberEngine, __k>` Class Template Reference

### Public Types

- `template<typename _Sseq>`  
`using _If_seed_seq = typename enable_if< __detail::__is_seed_seq< _Sseq, shuffle_order_engine, result_type>::value >::type`
- `typedef _RandomNumberEngine::result_type result_type`

### Public Member Functions

- `shuffle_order_engine ()`
- `shuffle_order_engine (const _RandomNumberEngine &__rng)`
- `shuffle_order_engine (_RandomNumberEngine &&__rng)`
- `shuffle_order_engine (result_type __s)`
- `template<typename _Sseq, typename = _If_seed_seq<_Sseq>>`  
`shuffle_order_engine (_Sseq &__q)`
- `const _RandomNumberEngine & base () const noexcept`
- `void discard (unsigned long long __z)`
- `result_type operator() ()`
- `void seed ()`
- `void seed (result_type __s)`
- `template<typename _Sseq>`  
`_If_seed_seq<_Sseq> seed (_Sseq &__q)`

### Static Public Member Functions

- `static constexpr result_type max ()`
- `static constexpr result_type min ()`

### Static Public Attributes

- `static constexpr size_t table_size`

### Friends

- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)`
- `bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)`
- `template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>`  
`std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)`



#### 4.933.1 Detailed Description

```
template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine< _RandomNumberEngine, __k >
```

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specifies number of bits \_\_k.

Definition at line 1324 of file random.h.

#### 4.933.2 Member Typedef Documentation

##### 4.933.2.1 result\_type

```
template<typename _RandomNumberEngine, size_t __k>
typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >↔
::result_type
```

The type of the generated random value.

Definition at line 1327 of file random.h.

#### 4.933.3 Constructor & Destructor Documentation

##### 4.933.3.1 shuffle\_order\_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine () [inline]
```

Constructs a default shuffle\_order\_engine engine.

The underlying engine is default constructed as well.

Definition at line 1344 of file random.h.

##### 4.933.3.2 shuffle\_order\_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 const _RandomNumberEngine & __rng) [inline], [explicit]
```

Copy constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1355 of file random.h.

**4.933.3.3 shuffle\_order\_engine()** [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 _RandomNumberEngine && __rng) [inline], [explicit]
```

Move constructs a shuffle\_order\_engine engine.

Copies an existing base class random number generator.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__rng</code> | An existing (base class) engine object. |
|--------------------|-----------------------------------------|

Definition at line 1366 of file random.h.

**4.933.3.4 shuffle\_order\_engine()** [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 result_type __s) [inline], [explicit]
```

Seed constructs a shuffle\_order\_engine engine.

Constructs the underlying generator engine seeded with \_\_s.

**Parameters**

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A seed value for the base class engine. |
|------------------|-----------------------------------------|

Definition at line 1377 of file random.h.

**4.933.3.5 shuffle\_order\_engine()** [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq , typename = _If_seed_seq<_Sseq>>
```

```
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
 _Sseq & __q) [inline], [explicit]
```

Generator construct a shuffle\_order\_engine engine.

#### Parameters

|                   |                  |
|-------------------|------------------|
| $\leftrightarrow$ | A seed sequence. |
| $q$               |                  |

Definition at line 1388 of file random.h.

### 4.933.4 Member Function Documentation

#### 4.933.4.1 base()

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine& std::shuffle_order_engine< _RandomNumberEngine, __k >::base () const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

Definition at line 1431 of file random.h.

#### 4.933.4.2 discard()

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 1452 of file random.h.

#### 4.933.4.3 max()

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max () [inline],
[static]
```

Gets the maximum value in the generated random number range.

Definition at line 1445 of file random.h.

References std::max().

#### 4.933.4.4 min()

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min () [inline],
[static]
```

Gets the minimum value in the generated random number range.

Definition at line 1438 of file random.h.

References std::min().

#### 4.933.4.5 operator()()

```
template<typename _RandomNumberEngine , size_t __k>
shuffle_order_engine< _RandomNumberEngine, __k >::result_type std::shuffle_order_engine< _↵
RandomNumberEngine, __k >::operator() ()
```

Gets the next value in the generated random number sequence.

Definition at line 810 of file bits/random.tcc.

#### 4.933.4.6 seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed () [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1397 of file random.h.

#### 4.933.4.7 seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
 result_type __s) [inline]
```

Reseeds the shuffle\_order\_engine object with the default seed for the underlying base class generator engine.

Definition at line 1408 of file random.h.

#### 4.933.4.8 seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq >
_If_seed_seq<_Sseq> std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
 _Sseq & __q) [inline]
```

Reseeds the shuffle\_order\_engine object with the given seed sequence.

## Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__q</code> | A seed generator function. |
|------------------|----------------------------|

Definition at line 1421 of file random.h.

## 4.933.5 Friends And Related Function Documentation

## 4.933.5.1 operator&lt;&lt;

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x) [friend]
```

Inserts the current state of a shuffle\_order\_engine random number generator engine \_\_x into the output stream \_\_os.

## Parameters

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| <code>__os</code> | An output stream.                                      |
| <code>__x</code>  | A shuffle_order_engine random number generator engine. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 4.933.5.2 operator==

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
 const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
 const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs) [friend]
```

Compares two shuffle\_order\_engine random number generator objects of the same type for equality.

## Parameters

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <code>__lhs</code> | A shuffle_order_engine random number generator object.       |
| <code>__rhs</code> | Another shuffle_order_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.



Definition at line 1476 of file random.h.

**4.933.5.3 operator>>**

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits > & __is,
 std::shuffle_order_engine<_RandomNumberEngine1, __k1 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

|                                                                                           |                                                        |
|-------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <br>__is | An input stream.                                       |
| <br>__x  | A shuffle_order_engine random number generator engine. |

**Returns**

The input stream with the state of \_\_x extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**4.934 std::slice Class Reference****Public Member Functions**

- [slice](#) ()
- [slice](#) (size\_t \_\_o, size\_t \_\_d, size\_t \_\_s)
- size\_t [size](#) () const
- size\_t [start](#) () const
- size\_t [stride](#) () const

## 4.934.1 Detailed Description

Class defining one-dimensional subset of an array.

The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.

For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

Definition at line 59 of file `slice_array.h`.

The documentation for this class was generated from the following file:

- [slice\\_array.h](#)

4.935 `std::slice_array<_Tp>` Class Template Reference

## Public Types

- typedef `_Tp` **value\_type**

## Public Member Functions

- [slice\\_array](#) (const [slice\\_array](#) &)
- void **operator%=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator%=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator&=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator&=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator\*=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator\*=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator+>=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator+>=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator->=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator->=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator/=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator/=>** (const \_Expr<\_Dom, \_Tp> &) const
- void **operator<=>** (const [valarray](#)<\_Tp> &) const
- template<class \_Dom>  
void **operator<=>** (const \_Expr<\_Dom, \_Tp> &) const
- [slice\\_array](#) & **operator=** (const [slice\\_array](#) &)
- void **operator=** (const [valarray](#)<\_Tp> &) const

- void [operator=](#) (const \_Tp &) const
- template<class \_Dom >  
void **operator=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator>>=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator>>=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator^=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator^=** (const \_Expr< \_Dom, \_Tp > &) const
- void [operator|=](#) (const [valarray](#)< \_Tp > &) const
- template<class \_Dom >  
void **operator|=** (const \_Expr< \_Dom, \_Tp > &) const

#### Friends

- class [valarray](#)< \_Tp >

#### 4.935.1 Detailed Description

```
template<typename _Tp>
class std::slice_array< _Tp >
```

Reference to one-dimensional subset of an array.

A slice\_array is a reference to the actual elements of an array specified by a slice. The way to get a slice\_array is to call operator[] (slice) on a valarray. The returned slice\_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice\_array refers to.

#### Parameters

|           |               |
|-----------|---------------|
| <i>Tp</i> | Element type. |
|-----------|---------------|

Definition at line 91 of file valarray.

The documentation for this class was generated from the following files:

- [valarray](#)
- [slice\\_array.h](#)

#### 4.936 \_\_gnu\_cxx::slist< \_Tp, \_Alloc > Class Template Reference

Inherits [\\_\\_gnu\\_cxx::Slist\\_base](#)< \_Tp, \_Alloc >.



## Public Types

- `typedef _Base::allocator_type allocator_type`
- `typedef _Slist_iterator< _Tp, const _Tp &, const _Tp * > const_iterator`
- `typedef const value_type * const_pointer`
- `typedef const value_type & const_reference`
- `typedef std::ptrdiff_t difference_type`
- `typedef _Slist_iterator< _Tp, _Tp &, _Tp * > iterator`
- `typedef value_type * pointer`
- `typedef value_type & reference`
- `typedef std::size_t size_type`
- `typedef _Tp value_type`

## Public Member Functions

- `slist (const allocator_type &__a=allocator_type())`
- `slist (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())`
- `slist (size_type __n)`
- `template<class _InputIterator >`  
`slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())`
- `slist (const slist &__x)`
- `template<class _Integer >`  
`void _M_assign_dispatch (_Integer __n, _Integer __val, std::__true_type)`
- `template<class _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, std::__false_type)`
- `void _M_fill_assign (size_type __n, const _Tp &__val)`
- `void assign (size_type __n, const _Tp &__val)`
- `template<class _InputIterator >`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `iterator before_begin ()`
- `const_iterator before_begin () const`
- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `iterator erase (iterator __pos)`
- `iterator erase (iterator __first, iterator __last)`
- `iterator erase_after (iterator __pos)`
- `iterator erase_after (iterator __before_first, iterator __last)`
- `reference front ()`
- `const_reference front () const`
- `allocator_type get_allocator () const`
- `iterator insert (iterator __pos, const value_type &__x)`
- `iterator insert (iterator __pos)`
- `void insert (iterator __pos, size_type __n, const value_type &__x)`
- `template<class _InIterator >`  
`void insert (iterator __pos, _InIterator __first, _InIterator __last)`
- `iterator insert_after (iterator __pos, const value_type &__x)`

- iterator **insert\_after** (iterator \_\_pos)
- void **insert\_after** (iterator \_\_pos, size\_type \_\_n, const value\_type &\_\_x)
- template<class \_InIterator >  
void **insert\_after** (iterator \_\_pos, \_InIterator \_\_first, \_InIterator \_\_last)
- size\_type **max\_size** () const
- void **merge** (slist &\_\_x)
- template<class \_StrictWeakOrdering >  
void **merge** (slist &, \_StrictWeakOrdering)
- slist & **operator=** (const slist &\_\_x)
- void **pop\_front** ()
- iterator **previous** (const\_iterator \_\_pos)
- const\_iterator **previous** (const\_iterator \_\_pos) const
- void **push\_front** (const value\_type &\_\_x)
- void **push\_front** ()
- void **remove** (const \_Tp &\_\_val)
- template<class \_Predicate >  
void **remove\_if** (\_Predicate \_\_pred)
- void **resize** (size\_type new\_size, const \_Tp &\_\_x)
- void **resize** (size\_type new\_size)
- void **reverse** ()
- size\_type **size** () const
- void **sort** ()
- template<class \_StrictWeakOrdering >  
void **sort** (\_StrictWeakOrdering \_\_comp)
- void **splice** (iterator \_\_pos, slist &\_\_x)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_i)
- void **splice** (iterator \_\_pos, slist &\_\_x, iterator \_\_first, iterator \_\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_before\_first, iterator \_\_before\_last)
- void **splice\_after** (iterator \_\_pos, iterator \_\_prev)
- void **splice\_after** (iterator \_\_pos, slist &\_\_x)
- void **swap** (slist &\_\_x)
- void **unique** ()
- template<class \_BinaryPredicate >  
void **unique** (\_BinaryPredicate \_\_pred)

#### Private Types

- typedef \_\_alloc\_traits<\_Alloc >::template rebind<\_Slist\_node<\_Tp > >::other \_Node\_alloc

#### Private Member Functions

- \_Slist\_node\_base \* **\_M\_erase\_after** (\_Slist\_node\_base \* \_\_pos)
- \_Slist\_node\_base \* **\_M\_erase\_after** (\_Slist\_node\_base \*, \_Slist\_node\_base \*)
- \_Slist\_node<\_Tp > \* **\_M\_get\_node** ()
- void **\_M\_put\_node** (\_Slist\_node<\_Tp > \* \_\_p)

#### Private Attributes

- \_Slist\_node\_base **\_M\_head**

## 4.936.1 Detailed Description

```
template<class _Tp, class _Alloc = std::allocator<_Tp>>
class __gnu_cxx::slist<_Tp, _Alloc >
```

This is an SGI extension.

**Todo** Needs documentation! See [http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<\\_style.html](http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation-<_style.html)

Definition at line 287 of file `slist`.

The documentation for this class was generated from the following file:

- `slist`

4.937 `__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >` Class Template Reference

Inherits `__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >`.

## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `base_type::const_pointer` **const\_pointer**
- typedef `base_type::const_reference` **const\_reference**
- typedef `base_type::const_reverse_iterator` **const\_reverse\_iterator**
- typedef `splay_tree_tag` **container\_category**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::key_const_pointer` **key\_const\_pointer**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_pointer` **key\_pointer**
- typedef `base_type::key_reference` **key\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `base_type::mapped_const_pointer` **mapped\_const\_pointer**
- typedef `base_type::mapped_const_reference` **mapped\_const\_reference**
- typedef `base_type::mapped_pointer` **mapped\_pointer**
- typedef `base_type::mapped_reference` **mapped\_reference**
- typedef `base_type::mapped_type` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `traits_type::node_const_iterator` **node\_const\_iterator**
- typedef `traits_type::node_iterator` **node\_iterator**

- typedef base\_type::node\_update **node\_update**
- typedef base\_type::const\_iterator **point\_const\_iterator**
- typedef base\_type::point\_iterator **point\_iterator**
- typedef base\_type::pointer **pointer**
- typedef base\_type::reference **reference**
- typedef base\_type::reverse\_iterator **reverse\_iterator**
- typedef \_Alloc::size\_type **size\_type**
- typedef integral\_constant< int, Store\_Hash > **store\_extra**
- typedef stored\_data< value\_type, size\_type, Store\_Hash > **stored\_data\_type**
- typedef base\_type::value\_type **value\_type**

#### Public Member Functions

- **splay\_tree\_map** (const Cmp\_Fn &)
- **splay\_tree\_map** (const Cmp\_Fn &, const node\_update &)
- **splay\_tree\_map** (const splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- iterator **begin** ()
- const\_iterator **begin** () const
- void **clear** ()
- template<typename It >  
void **copy\_from\_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const\_iterator **end** () const
- bool **erase** (key\_const\_reference)
- iterator **erase** (iterator it)
- reverse\_iterator **erase** (reverse\_iterator)
- template<typename Pred >  
size\_type **erase\_if** (Pred)
- point\_iterator **find** (key\_const\_reference)
- point\_const\_iterator **find** (key\_const\_reference) const
- Cmp\_Fn & **get\_cmp\_fn** ()
- const Cmp\_Fn & **get\_cmp\_fn** () const
- void **initialize** ()
- std::pair< point\_iterator, bool > **insert** (const\_reference r\_value)
- void **join** (splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **lower\_bound** (key\_const\_reference)
- point\_const\_iterator **lower\_bound** (key\_const\_reference) const
- size\_type **max\_size** () const
- node\_const\_iterator **node\_begin** () const
- node\_iterator **node\_begin** ()
- node\_const\_iterator **node\_end** () const
- node\_iterator **node\_end** ()
- mapped\_reference **operator[]** (key\_const\_reference r\_key)
- reverse\_iterator **rbegin** ()
- const\_reverse\_iterator **rbegin** () const
- reverse\_iterator **rend** ()
- const\_reverse\_iterator **rend** () const
- size\_type **size** () const
- void **split** (key\_const\_reference, splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- point\_iterator **upper\_bound** (key\_const\_reference)
- point\_const\_iterator **upper\_bound** (key\_const\_reference) const

### Public Attributes

- no\_throw\_indicator **m\_no\_throw\_copies\_indicator**
- store\_extra **m\_store\_extra\_indicator**

### Protected Types

- typedef node\_alloc\_traits::value\_type **node**
- typedef node\_alloc\_traits::allocator\_type **node\_allocator**
- typedef traits\_type::null\_node\_update\_pointer **null\_node\_update\_pointer**
- typedef [types\\_traits](#)< Key, Mapped, \_Alloc, false > **traits\_base**

### Protected Member Functions

- void **actual\_erase\_node** (node\_pointer)
- void **apply\_update** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **apply\_update** (node\_pointer, Node\_Update\_\*)
- [std::pair](#)< node\_pointer, bool > **erase** (node\_pointer)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, false\_type)
- node\_pointer **get\_new\_node\_for\_leaf\_insert** (const\_reference, true\_type)
- void **initialize\_min\_max** ()
- iterator **insert\_imp\_empty** (const\_reference)
- [std::pair](#)< point\_iterator, bool > **insert\_leaf** (const\_reference)
- iterator **insert\_leaf\_new** (const\_reference, node\_pointer, bool)
- void **join\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **join\_prep** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- size\_type **recursive\_count** (node\_pointer) const
- void **rotate\_left** (node\_pointer)
- void **rotate\_parent** (node\_pointer)
- void **rotate\_right** (node\_pointer)
- void **split\_finish** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- bool **split\_prep** (key\_const\_reference, bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)
- void **update\_min\_max\_for\_erased\_node** (node\_pointer)
- void **update\_to\_top** (node\_pointer, null\_node\_update\_pointer)
- template<typename Node\_Update\_>  
void **update\_to\_top** (node\_pointer, Node\_Update\_\*)
- void **value\_swap** (bin\_search\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc > &)

### Static Protected Member Functions

- static void **clear\_imp** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_head**
- size\_type **m\_size**

## Static Protected Attributes

- static node\_allocator **s\_node\_allocator**

### 4.937.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
```

Splay tree.

Definition at line 107 of file splay\_tree\_.hpp.

### 4.937.2 Member Function Documentation

#### 4.937.2.1 node\_begin() [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
begin () const [inline], [inherited]
```

Returns a const node\_iterator corresponding to the node at the root of the tree.

Definition at line 111 of file bin\_search\_tree\_.hpp.

#### 4.937.2.2 node\_begin() [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a node\_iterator corresponding to the node at the root of the tree.

Definition at line 119 of file bin\_search\_tree\_.hpp.

#### 4.937.2.3 `node_end()` [1/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↵
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↵
end () const [inline], [inherited]
```

Returns a const `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 127 of file `bin_search_tree_.hpp`.

#### 4.937.2.4 `node_end()` [2/2]

```
template<typename Key , typename Mapped , typename Cmp_Fn , typename Node_And_It_Traits , typename
_Alloc >
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↵
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end (
) [inline], [inherited]
```

Returns a `node_iterator` corresponding to a node just after a leaf of the tree.

Definition at line 135 of file `bin_search_tree_.hpp`.

The documentation for this class was generated from the following file:

- [splay\\_tree\\_.hpp](#)

## 4.938 `__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >` Struct Template Reference

### Public Types

- typedef `rebind_traits< _Alloc, metadata_type >::const_reference` **metadata\_const\_reference**
- typedef `rebind_traits< _Alloc, metadata_type >::reference` **metadata\_reference**
- typedef `Metadata` **metadata\_type**
- typedef `rebind_traits< _Alloc, splay_tree_node_ >::pointer` **node\_pointer**
- typedef `Value_Type` **value\_type**

### Public Member Functions

- `metadata_const_reference` **get\_metadata** () const
- `metadata_reference` **get\_metadata** ()
- `bool` **special** () const

### Public Attributes

- metadata\_type **m\_metadata**
- node\_pointer **m\_p\_left**
- node\_pointer **m\_p\_parent**
- node\_pointer **m\_p\_right**
- bool **m\_special**
- value\_type **m\_value**

#### 4.938.1 Detailed Description

```
template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_ < Value_Type, Metadata, _Alloc >
```

Node for splay tree.

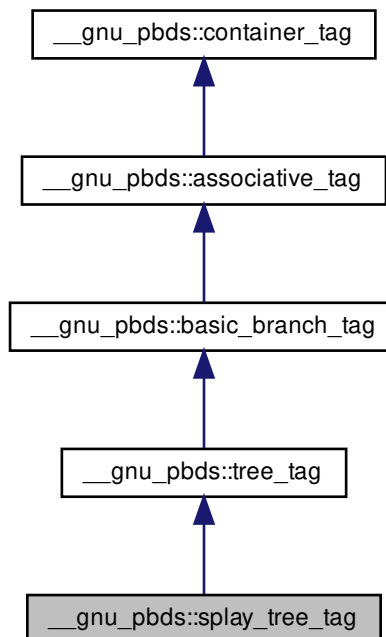
Definition at line 50 of file splay\_tree\_/node.hpp.

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/node.hpp](#)

#### 4.939 \_\_gnu\_pbds::splay\_tree\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::splay\_tree\_tag:





## 4.939.1 Detailed Description

Splay tree.

Definition at line 156 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.940 `std::stack<_Tp, _Sequence>` Class Template Reference

## Public Types

- `typedef _Sequence::const_reference` **const\_reference**
- `typedef _Sequence` **container\_type**
- `typedef _Sequence::reference` **reference**
- `typedef _Sequence::size_type` **size\_type**
- `typedef _Sequence::value_type` **value\_type**

## Public Member Functions

- `template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>`  
`stack()`
- `stack(const _Sequence &__c)`
- `stack(_Sequence &&__c)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack(const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack(const _Sequence &__c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack(_Sequence &&__c, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack(const stack &__q, const _Alloc &__a)`
- `template<typename _Alloc, typename _Requires = _Uses<_Alloc>>`  
`stack(stack &&__q, const _Alloc &__a)`
- `template<typename... _Args>`  
`void emplace (_Args &&... __args)`
- `bool empty() const`
- `void pop()`
- `void push(const value_type &__x)`
- `void push(value_type &&__x)`
- `size_type size() const`
- `void swap(stack &__s) noexcept(__is_nothrow_swappable<_Sequence>::value)`
- `reference top()`
- `const_reference top() const`

## Protected Attributes

- `_Sequence c`

## Friends

- `template<typename _Tp1, typename _Seq1 >`  
`bool operator< (const stack<_Tp1, _Seq1 > &, const stack<_Tp1, _Seq1 > &)`
- `template<typename _Tp1, typename _Seq1 >`  
`bool operator== (const stack<_Tp1, _Seq1 > &, const stack<_Tp1, _Seq1 > &)`

### 4.940.1 Detailed Description

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::stack<_Tp, _Sequence >
```

A standard container giving FILO behavior.

#### Template Parameters

|                        |                                                                          |
|------------------------|--------------------------------------------------------------------------|
| <code>_Tp</code>       | Type of element.                                                         |
| <code>_Sequence</code> | Type of underlying sequence, defaults to <code>deque&lt;_Tp&gt;</code> . |

Meets many of the requirements of a [container](#), but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to `std::deque`, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as `std::list`, `std::vector`, or an appropriate user-defined type.

Members not found in *normal* containers are `container_type`, which is a typedef for the second `Sequence` parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

Definition at line 99 of file `stl_stack.h`.

### 4.940.2 Constructor & Destructor Documentation

#### 4.940.2.1 `stack()`

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↔
_Seq>::value>::type>
std::stack<_Tp, _Sequence >::stack () [inline]
```

Default constructor creates no elements.

Definition at line 162 of file `stl_stack.h`.

### 4.940.3 Member Function Documentation

#### 4.940.3.1 empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack<_Tp, _Sequence >::empty () const [inline]
```

Returns true if the stack is empty.

Definition at line 199 of file stl\_stack.h.

#### 4.940.3.2 pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence >::pop () [inline]
```

Removes first element.

This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.


Definition at line 272 of file stl\_stack.h.

#### 4.940.3.3 push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack<_Tp, _Sequence >::push (
 const value_type & __x) [inline]
```

Add data to the top of the stack.

##### Parameters

|                                                                                        |                   |
|----------------------------------------------------------------------------------------|-------------------|
|  _x | Data to be added. |
|----------------------------------------------------------------------------------------|-------------------|

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

Definition at line 239 of file stl\_stack.h.

#### 4.940.3.4 size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack<_Tp, _Sequence >::size () const [inline]
```

Returns the number of elements in the stack.

Definition at line 204 of file `stl_stack.h`.

#### 4.940.3.5 top() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack<_Tp, _Sequence >::top () [inline]
```

Returns a read/write reference to the data at the first element of the stack.

Definition at line 212 of file `stl_stack.h`.

#### 4.940.3.6 top() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack<_Tp, _Sequence >::top () const [inline]
```

Returns a read-only (constant) reference to the data at the first element of the stack.

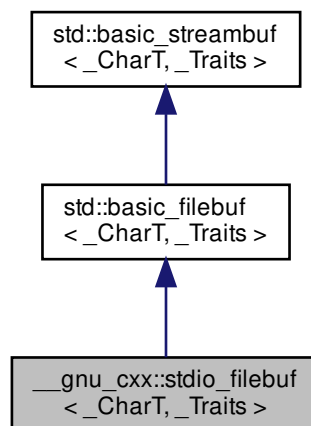
Definition at line 223 of file `stl_stack.h`.

The documentation for this class was generated from the following file:

- [stl\\_stack.h](#)

### 4.941 \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits> Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_filebuf<_CharT, _Traits>`:



## Public Types

- `typedef codecvt< char_type, char, __state_type > __codecvt_type`
- `typedef __basic_file< char > __file_type`
- `typedef basic_filebuf< char_type, traits_type > __filebuf_type`
- `typedef traits_type::state_type __state_type`
- `typedef basic_streambuf< char_type, traits_type > __streambuf_type`
- `typedef _CharT char_type`
- `typedef traits_type::int_type int_type`
- `typedef traits_type::off_type off_type`
- `typedef traits_type::pos_type pos_type`
- `typedef std::size_t size_t`
- `typedef _Traits traits_type`

## Public Member Functions

- `stdio_filebuf ()`
  - `stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
  - `stdio_filebuf (std::c_file * __f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))`
  - `stdio_filebuf (stdio_filebuf &&)=default`
  - `virtual ~stdio_filebuf ()`
  - `__filebuf_type * close ()`
  - `int fd ()`
  - `std::c_file * file ()`
  - `locale getloc () const`
  - `streamsize in_avail ()`
  - `bool is_open () const throw ()`
  - `__filebuf_type * open (const char * __s, ios_base::openmode __mode)`
  - `__filebuf_type * open (const std::string & __s, ios_base::openmode __mode)`
  - `stdio_filebuf & operator= (stdio_filebuf &&)=default`
  - `locale pubimbue (const locale & __loc)`
  - `int_type sbumpc ()`
  - `int_type sgetc ()`
  - `streamsize sgetn (char_type * __s, streamsize __n)`
  - `int_type snextc ()`
  - `int_type sputbackc (char_type __c)`
  - `int_type sputc (char_type __c)`
  - `streamsize sputn (const char_type * __s, streamsize __n)`
  - `int_type sungetc ()`
  - `void swap (stdio_filebuf & __fb)`
  - `void swap (basic_filebuf &)`
- 
- `basic_streambuf * pubsetbuf (char_type * __s, streamsize __n)`
  - `pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)`
  - `int pubsync ()`

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
  - void **\_\_safe\_pbump** (streamsize \_\_n)
  - void **\_M\_allocate\_internal\_buffer** ()
  - bool **\_M\_convert\_to\_external** (char\_type \*, streamsize)
  - void **\_M\_create\_pback** ()
  - void **\_M\_destroy\_internal\_buffer** () throw ()
  - void **\_M\_destroy\_pback** () throw ()
  - int **\_M\_get\_ext\_pos** (\_\_state\_type &\_\_state)
  - pos\_type **\_M\_seek** (off\_type \_\_off, ios\_base::seekdir \_\_way, \_\_state\_type \_\_state)
  - void **\_M\_set\_buffer** (streamsize \_\_off)
  - bool **\_M\_terminate\_output** ()
  - void **gbump** (int \_\_n)
  - virtual void **imbue** (const locale &\_\_loc)
  - virtual int\_type **overflow** (int\_type \_\_c=\_Traits::eof())
  - virtual int\_type **pbackfail** (int\_type \_\_c=\_Traits::eof())
  - void **pbump** (int \_\_n)
  - virtual pos\_type **seekoff** (off\_type \_\_off, ios\_base::seekdir \_\_way, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual pos\_type **seekpos** (pos\_type \_\_pos, ios\_base::openmode \_\_mode=ios\_base::in|ios\_base::out)
  - virtual **\_\_streambuf\_type** \* **setbuf** (char\_type \*\_\_s, streamsize \_\_n)
  - void **setg** (char\_type \*\_\_gbeg, char\_type \*\_\_gnext, char\_type \*\_\_gend)
  - void **setp** (char\_type \*\_\_pbeg, char\_type \*\_\_pend)
  - virtual streamsize **showmanyc** ()
  - void **swap** (basic\_streambuf &\_\_sb)
  - virtual int **sync** ()
  - virtual int\_type **uflow** ()
  - virtual int\_type **underflow** ()
  - virtual streamsize **xsgetn** (char\_type \*\_\_s, streamsize \_\_n)
  - virtual streamsize **xspu** (const char\_type \*\_\_s, streamsize \_\_n)
- 
- char\_type \* **eback** () const
  - char\_type \* **gptr** () const
  - char\_type \* **egptr** () const
- 
- char\_type \* **pbase** () const
  - char\_type \* **pptr** () const
  - char\_type \* **epptr** () const

## Protected Attributes

- `char_type * _M_buf`
  - `bool _M_buf_allocated`
  - `locale _M_buf_locale`
  - `size_t _M_buf_size`
  - `const __codecvt_type * _M_codecvt`
  - `char * _M_ext_buf`
  - `streamsize _M_ext_buf_size`
  - `char * _M_ext_end`
  - `const char * _M_ext_next`
  - `__file_type _M_file`
  - `char_type * _M_in_beg`
  - `char_type * _M_in_cur`
  - `char_type * _M_in_end`
  - `__c_lock _M_lock`
  - `ios_base::openmode _M_mode`
  - `char_type * _M_out_beg`
  - `char_type * _M_out_cur`
  - `char_type * _M_out_end`
  - `bool _M_reading`
  - `__state_type _M_state_beg`
  - `__state_type _M_state_cur`
  - `__state_type _M_state_last`
  - `bool _M_writing`
- 
- `char_type _M_pback`
  - `char_type * _M_pback_cur_save`
  - `char_type * _M_pback_end_save`
  - `bool _M_pback_init`

## 4.941.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C/POSIX.

This GNU extension provides extensions for working with standard C FILE\*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 50 of file `stdio_filebuf.h`.

## 4.941.2 Constructor &amp; Destructor Documentation

**4.941.2.1 stdio\_filebuf()** [1/3]

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf () [inline]
```

deferred initialization

Definition at line 65 of file `stdio_filebuf.h`.

**4.941.2.2 stdio\_filebuf()** [2/3]

```
template<typename _CharT , typename _Traits >
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
 int __fd,
 std::ios_base::openmode __mode,
 size_t __size = static_cast<size_t>(BUFSIZ))
```

**Parameters**

|                     |                                                         |
|---------------------|---------------------------------------------------------|
| <code>__fd</code>   | An open file descriptor.                                |
| <code>__mode</code> | Same meaning as in a standard filebuf.                  |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. |

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the `stdio_filebuf` is closed/destroyed.

Definition at line 137 of file `stdio_filebuf.h`.

**4.941.2.3 stdio\_filebuf()** [3/3]

```
template<typename _CharT , typename _Traits >
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
 std::__c_file * __f,
 std::ios_base::openmode __mode,
 size_t __size = static_cast<size_t>(BUFSIZ))
```

**Parameters**

|                     |                                                                                                    |
|---------------------|----------------------------------------------------------------------------------------------------|
| <code>__f</code>    | An open <code>FILE*</code> .                                                                       |
| <code>__mode</code> | Same meaning as in a standard filebuf.                                                             |
| <code>__size</code> | Optimal or preferred size of internal buffer, in chars. Defaults to system's <code>BUFSIZ</code> . |

This constructor associates a file stream buffer with an open C `FILE*`. The `FILE*` will not be automatically closed when the `stdio_filebuf` is closed/destroyed.



Definition at line 153 of file `stdio_filebuf.h`.

#### 4.941.2.4 `~stdio_filebuf()`

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf<_CharT, _Traits>::~~stdio_filebuf () [virtual]
```

Closes the external data stream if the file descriptor constructor was used.

Definition at line 132 of file `stdio_filebuf.h`.

### 4.941.3 Member Function Documentation

#### 4.941.3.1 `_M_create_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_create_pback () [inline], [protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: `_M_in_cur` has already been moved back

Definition at line 199 of file `fstream`.

#### 4.941.3.2 `_M_destroy_pback()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_destroy_pback () throw () [inline], [protected],
[inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

Definition at line 216 of file `fstream`.

#### 4.941.3.3 `_M_set_buffer()`

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf<_CharT, _Traits>::_M_set_buffer (
 streamsize __off) [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:

`__off == egptr() - eback()` upon underflow/uflow (**read** mode); `__off == 0` upon overflow (**write** mode); `__off == -1` upon open, setbuf, seekoff/pos (**uncommitted** mode).

NB: `egptr() - pbase() == _M_buf_size - 1`, since `_M_buf_size` reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

Definition at line 459 of file `fstream`.

Referenced by `std::basic_filebuf<char_type, traits_type>::close()`.

#### 4.941.3.4 close()

```
template<typename _CharT, typename _Traits >
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
() [inherited]
```

Closes the currently associated file.

##### Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.

If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.

If any operations fail, this function also fails.

Definition at line 250 of file `fstream.tcc`.

#### 4.941.3.5 eback()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 4.941.3.6 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

#### 4.941.3.7 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 4.941.3.8 fd()

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf<_CharT, _Traits>::fd () [inline]
```

##### Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

Definition at line 118 of file stdio\_filebuf.h.

#### 4.941.3.9 file()

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
std::__c_file* __gnu_cxx::stdio_filebuf<_CharT, _Traits>::file () [inline]
```

##### Returns

The underlying FILE\*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 128 of file stdio\_filebuf.h.

#### 4.941.3.10 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**4.941.3.11 getloc()**

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**4.941.3.12 gptr()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**4.941.3.13 imbue()**

```
template<typename _CharT , typename _Traits >
void std::basic_filebuf< _CharT, _Traits >::imbue (
 const locale & __loc) [protected], [virtual], [inherited]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 1030 of file fstream.tcc.

## 4.941.3.14 in\_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

## 4.941.3.15 is\_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf<_CharT, _Traits>::is_open () const throw () [inline], [inherited]
```

Returns true if the external file is open.

Definition at line 265 of file fstream.

## 4.941.3.16 open() [1/2]

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::__filebuf_type * std::basic_filebuf<_CharT, _Traits>::open (
 const char * __s,
 ios_base::openmode __mode) [inherited]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, `NULL` on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named `__s` using the flags given in `__mode`.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines `app`, `in|out|app`, `in|app`, `binary|app`, `binary|in|out|app`, and `binary|in|app` per DR 596)

| ios_base Flag combination |    |     |       |     | stdio equivalent |
|---------------------------|----|-----|-------|-----|------------------|
| binary                    | in | out | trunc | app |                  |
|                           |    | +   |       |     | w                |
|                           |    | +   |       | +   | a                |
|                           |    |     |       | +   | a                |
|                           |    | +   | +     |     | w                |
|                           | +  |     |       |     | r                |
|                           | +  | +   |       |     | r+               |
|                           | +  | +   | +     |     | w+               |
|                           | +  | +   |       | +   | a+               |
|                           | +  |     |       | +   | a+               |
| +                         |    | +   |       |     | wb               |
| +                         |    | +   |       | +   | ab               |
| +                         |    |     |       | +   | ab               |
| +                         |    | +   | +     |     | wb               |
| +                         | +  |     |       |     | rb               |
| +                         | +  | +   |       |     | r+b              |
| +                         | +  | +   | +     |     | w+b              |
| +                         | +  | +   |       | +   | a+b              |
| +                         | +  |     |       | +   | a+b              |

Definition at line 180 of file `fstream.tcc`.

Referenced by `std::basic_filebuf< char_type, traits_type >::open()`.

4.941.3.17 `open()` [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type* std::basic_filebuf< _CharT, _Traits >::open (
 const std::string & __s,
 ios_base::openmode __mode) [inline], [inherited]
```

Opens an external file.

## Parameters

|                     |                       |
|---------------------|-----------------------|
| <code>__s</code>    | The name of the file. |
| <code>__mode</code> | The open mode flags.  |

## Returns

`this` on success, `NULL` on failure

Definition at line 331 of file `fstream`.

4.941.3.18 `overflow()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

## Returns

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

## Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 542 of file `fstream.tcc`.

4.941.3.19 `pbackfail()`

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
 int_type __c = _Traits::eof()) [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                 |                                                      |
|-----------------|------------------------------------------------------|
| <code>_c</code> | The character to be inserted back into the sequence. |
|-----------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 483 of file `fstream.tcc`.

**4.941.3.20 pbase()**

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `eptr()` returns the end pointer for the output sequence

Definition at line 536 of file `streambuf`.

**4.941.3.21 pbump()**

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.



## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the write position without returning any data.

Definition at line 552 of file `streambuf`.

4.941.3.22 `pptr()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 539 of file `streambuf`.

4.941.3.23 `pubimbue()`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for `imbue()`.

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls the derived `imbue(__loc)`.

Definition at line 216 of file `streambuf`.

#### 4.941.3.24 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

##### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__off</code>  | Offset.                                     |
| <code>__way</code>  | Value for <code>ios_base::seekdir</code> .  |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 4.941.3.25 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

##### Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual seekpos function.

Definition at line 270 of file streambuf.

#### 4.941.3.26 pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

#### 4.941.3.27 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]
```

Calls virtual sync function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::sync()`.

#### 4.941.3.28 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]
```

Getting the next character.

#### Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.941.3.29 `seekoff()`

```
template<typename _CharT, typename _Traits>
basic_filebuf<_CharT, _Traits>::pos_type std::basic_filebuf<_CharT, _Traits>::seekoff (
 off_type,
 ios_base::seekdir,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

#### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 833 of file `fstream.tcc`.

**4.941.3.30 seekpos()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [protected], [virtual], [inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

**Note**

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 893 of file `fstream.tcc`.

**4.941.3.31 setbuf()**

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↵
::setbuf (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Manipulates the buffer.

**Parameters**

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__s</code> | Pointer to a buffer area.  |
| <code>↵<br/>__n</code> | Size of <code>__s</code> . |

**Returns**

`this`

If no file has been opened, and both `__s` and `__n` are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↵html#io.streambuf.buffering> for more.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 804 of file `fstream.tcc`.

## 4.941.3.32 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

## 4.941.3.33 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

## Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

**4.941.3.34 sgetc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

**4.941.3.35 sgetn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**4.941.3.36 showmanyc()**

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc () [protected], [virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail.* [27.5.2.4.3]/1

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 298 of file `fstream.tcc`.

**4.941.3.37 `snextc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**4.941.3.38 `sputbackc()`**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.941.3.39 sputc()**

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↵__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator< _CharT, _Traits >::operator=()`.

**4.941.3.40 sputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.



## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xspn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

4.941.3.41 `sungetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

## Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbackfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

4.941.3.42 `sync()`

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf<_CharT, _Traits>::sync () [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

## Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

## Note

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 1013 of file `fstream.tcc`.

#### 4.941.3.43 uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow () [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented in `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`.

Definition at line 707 of file `streambuf`.

#### 4.941.3.44 underflow()

```
template<typename _CharT , typename _Traits >
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 324 of file `fstream.tcc`.

#### 4.941.3.45 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 670 of file `fstream.tcc`.

4.941.3.46 `xspn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf<_CharT, _Traits>::xspn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 756 of file `fstream.tcc`.

#### 4.941.4 Member Data Documentation

##### 4.941.4.1 `_M_buf`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf [protected], [inherited]
```

Pointer to the beginning of internal buffer.

Definition at line 136 of file `fstream`.

##### 4.941.4.2 `_M_buf_locale`

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file `streambuf`.

##### 4.941.4.3 `_M_buf_size`

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

Definition at line 143 of file `fstream`.

##### 4.941.4.4 `_M_ext_buf`

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf [protected], [inherited]
```

Buffer for external characters. Used for input when `codecvt::always_noconv() == false`. When valid, this corresponds to `eback()`.

Definition at line 178 of file `fstream`.

#### 4.941.4.5 \_M\_ext\_buf\_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size [protected], [inherited]
```

Size of buffer held by \_M\_ext\_buf.

Definition at line 183 of file fstream.

#### 4.941.4.6 \_M\_ext\_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next [protected], [inherited]
```

Pointers into the buffer held by \_M\_ext\_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, \_M\_ext\_next corresponds to egptr().

Definition at line 190 of file fstream.

#### 4.941.4.7 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 4.941.4.8 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

#### 4.941.4.9 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 4.941.4.10 `_M_mode`

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.

Definition at line 121 of file fstream.

Referenced by `std::basic_filebuf`< char\_type, traits\_type >::close().

#### 4.941.4.11 `_M_out_beg`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.941.4.12 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.941.4.13 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

#### 4.941.4.14 \_M\_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf<_CharT, _Traits>::_M_pback [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 164 of file fstream.

#### 4.941.4.15 \_M\_pback\_cur\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf<_CharT, _Traits>::_M_pback_cur_save [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 165 of file fstream.

#### 4.941.4.16 \_M\_pback\_end\_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf<_CharT, _Traits>::_M_pback_end_save [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 166 of file fstream.

#### 4.941.4.17 `_M_pback_init`

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init [protected], [inherited]
```

Necessary bits for putback buffer management.

##### Note

pbacks of over one character are not currently supported.

Definition at line 167 of file `fstream`.

Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

#### 4.941.4.18 `_M_reading`

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading [protected], [inherited]
```

`_M_reading == false` && `_M_writing == false` for **uncommitted** mode; `_M_reading == true` for **read** mode; `_M_writing == true` for **write** mode;

NB: `_M_reading == true` && `_M_writing == true` is unused.

Definition at line 155 of file `fstream`.

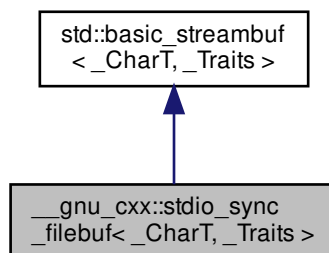
Referenced by `std::basic_filebuf< char_type, traits_type >::close()`.

The documentation for this class was generated from the following file:

- [stdio\\_filebuf.h](#)

### 4.942 `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >` Class Template Reference

Inheritance diagram for `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`:





## Public Types

- typedef `_CharT` **char\_type**
- typedef `traits_type::int_type` **int\_type**
- typedef `traits_type::off_type` **off\_type**
- typedef `traits_type::pos_type` **pos\_type**
- typedef `_Traits` **traits\_type**

## Public Member Functions

- **stdio\_sync\_filebuf** (`std::__c_file * __f`)
- **stdio\_sync\_filebuf** (`stdio_sync_filebuf && __fb`) noexcept
- `std::__c_file * file` ()
- locale `getloc` () const
- streamsize `in_avail` ()
- **stdio\_sync\_filebuf** & **operator=** (`stdio_sync_filebuf && __fb`) noexcept
- locale `pubimbue` (const locale & \_\_loc)
- `int_type sbumpc` ()
- `int_type sgetc` ()
- streamsize `sgetn` (`char_type * __s`, streamsize \_\_n)
- `int_type snextc` ()
- `int_type sputbackc` (`char_type __c`)
- `int_type sputc` (`char_type __c`)
- streamsize `sputn` (const `char_type * __s`, streamsize \_\_n)
- `int_type sungetc` ()
- void **swap** (`stdio_sync_filebuf & __fb`)
- `basic_streambuf * pubsetbuf` (`char_type * __s`, streamsize \_\_n)
- `pos_type pubseekoff` (`off_type __off`, `ios_base::seekdir __way`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- `pos_type pubseekpos` (`pos_type __sp`, `ios_base::openmode __mode=ios_base::in|ios_base::out`)
- int `pubsync` ()

## Protected Member Functions

- void **\_\_safe\_gbump** (streamsize \_\_n)
- void **\_\_safe\_pbump** (streamsize \_\_n)
- void `gbump` (int \_\_n)
- virtual void `imbue` (const locale & \_\_loc)
- virtual `int_type overflow` (`int_type __c=traits_type::eof()`)
- virtual `int_type pbackfail` (`int_type __c=traits_type::eof()`)
- void `pbump` (int \_\_n)
- virtual `std::streampos seekoff` (`std::streamoff __off`, `std::ios_base::seekdir __dir`, `std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out`)
- virtual `pos_type seekoff` (`off_type`, `ios_base::seekdir`, `ios_base::openmode=ios_base::in|ios_base::out`)
- virtual `std::streampos seekpos` (`std::streampos __pos`, `std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out`)
- virtual `pos_type seekpos` (`pos_type`, `ios_base::openmode=ios_base::in|ios_base::out`)

- virtual `basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)`
- void `setg (char_type * __gbeg, char_type * __gnext, char_type * __gend)`
- void `setp (char_type * __pbeg, char_type * __pend)`
- virtual streamsize `showmanyc ()`
- void `swap (basic_streambuf & __sb)`
- virtual int `sync ()`
- `int_type syncgetc ()`
- template<>  
`stdio_sync_filebuf< char >::int_type syncgetc ()`
- template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncgetc ()`
- `int_type syncputc (int_type __c)`
- template<>  
`stdio_sync_filebuf< char >::int_type syncputc (int_type __c)`
- template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncputc (int_type __c)`
- `int_type syncungetc (int_type __c)`
- template<>  
`stdio_sync_filebuf< char >::int_type syncungetc (int_type __c)`
- template<>  
`stdio_sync_filebuf< wchar_t >::int_type syncungetc (int_type __c)`
- virtual `int_type uflow ()`
- virtual `int_type underflow ()`
- virtual `std::streamsize xsgetn (char_type * __s, std::streamsize __n)`
- template<>  
`std::streamsize xsgetn (char * __s, std::streamsize __n)`
- template<>  
`std::streamsize xsgetn (wchar_t * __s, std::streamsize __n)`
- virtual streamsize `xsgetn (char_type * __s, streamsize __n)`
- virtual `std::streamsize xsputn (const char_type * __s, std::streamsize __n)`
- template<>  
`std::streamsize xsputn (const char * __s, std::streamsize __n)`
- template<>  
`std::streamsize xsputn (const wchar_t * __s, std::streamsize __n)`
- virtual streamsize `xsputn (const char_type * __s, streamsize __n)`

- `char_type * eback () const`
- `char_type * gptr () const`
- `char_type * egptr () const`

- `char_type * pbase () const`
- `char_type * pptr () const`
- `char_type * epptr () const`

### Protected Attributes

- locale `_M_buf_locale`
- `char_type` \* `_M_in_beg`
- `char_type` \* `_M_in_cur`
- `char_type` \* `_M_in_end`
- `char_type` \* `_M_out_beg`
- `char_type` \* `_M_out_cur`
- `char_type` \* `_M_out_end`

#### 4.942.1 Detailed Description

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>
```

Provides a layer of compatibility for C.

This GNU extension provides extensions for working with standard C FILE\*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., `stdio_filebuf<char>`.

Definition at line 57 of file `stdio_sync_filebuf.h`.

#### 4.942.2 Member Function Documentation

##### 4.942.2.1 `eback()`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 4.942.2.2 egptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Definition at line 495 of file streambuf.

#### 4.942.2.3 epptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 542 of file streambuf.

#### 4.942.2.4 file()

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
std::__c_file* __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file () [inline]
```

##### Returns

The underlying FILE\*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

Definition at line 118 of file stdio\_sync\_filebuf.h.

#### 4.942.2.5 gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

## Parameters

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

## 4.942.2.6 getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::getloc () const [inline], [inherited]
```

Locale access.

## Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

## 4.942.2.7 gptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

## 4.942.2.8 imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf<_CharT, _Traits>::imbue (
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

**Parameters**

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

**Note**

Base class version does nothing.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 583 of file streambuf.

**4.942.2.9 in\_avail()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

**Returns**

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

**4.942.2.10 overflow()**

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::overflow (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

**Parameters**

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 171 of file `stdio_sync_filebuf.h`.

**4.942.2.11 pbackfail()**

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_CharT, \\_Traits>](#).

Definition at line 146 of file `stdio_sync_filebuf.h`.

#### 4.942.2.12 pbase()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 536 of file streambuf.

#### 4.942.2.13 pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

##### Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

#### 4.942.2.14 pptr()

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.



## 4.942.2.15 pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

## Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

## Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

## 4.942.2.16 pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

## 4.942.2.17 pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
pos_type __sp,
ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

4.942.2.18 `pubsetbuf()`

```
template<typename _CharT, typename _Traits>
basic_streambuf* std::basic_streambuf<_CharT, _Traits>::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

4.942.2.19 `pubsync()`

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf<_CharT, _Traits>::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

Referenced by `std::wbuffer_convert<_Codecvt, _Elem, _Tr>::sync()`.

4.942.2.20 `sbumpc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sbumpc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.942.2.21 seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 609 of file `streambuf`.

#### 4.942.2.22 seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in `std::basic_filebuf< _CharT, _Traits >`, `std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >`, `std::basic_filebuf< char_type, traits_type >`, and `std::basic_stringbuf< _CharT, _Traits, _Alloc >`.

Definition at line 621 of file `streambuf`.

## 4.942.2.23 setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf<char_type, _Traits>* std::basic_streambuf<_CharT, _Traits>::setbuf (
 char_type * ,
 streamsize) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

## Note

Base class version does nothing, returns `this`.

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 598 of file `streambuf`.

## 4.942.2.24 setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf<_CharT, _Traits>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file `streambuf`.

#### 4.942.2.25 setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

##### Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

##### Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

#### 4.942.2.26 sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc () [inline], [inherited]
```

Getting the next character.

##### Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file streambuf.

#### 4.942.2.27 sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsggetn`.

## Parameters

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s, __n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

4.942.2.28 `showmanyc()`

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf<_CharT, _Traits>::showmanyc () [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

## Returns

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

## Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Reimplemented in `std::basic_filebuf<_CharT, _Traits>`, `std::basic_filebuf<_CharT, encoding_char_traits<_CharT>>`, `std::basic_filebuf<char_type, traits_type>`, and `std::basic_stringbuf<_CharT, _Traits, _Alloc>`.

Definition at line 656 of file `streambuf`.

**4.942.2.29** `snextc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

**4.942.2.30** `sputbackc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

**4.942.2.31** `sputc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.



## Parameters

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

## Returns

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

Referenced by `std::ostreambuf_iterator<_CharT, _Traits>::operator=()`.

4.942.2.32 `sputn()`

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

## Parameters

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xsgputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

4.942.2.33 `sungetc()`

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf<_CharT, _Traits>::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns `pbckfail()`. The effect is to *unget* the last character *gotten*.

Definition at line 404 of file `streambuf`.

**4.942.2.34 sync()**

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::sync (
 void) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 190 of file `stdio_sync_filebuf.h`.

**4.942.2.35 uflow()**

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::uflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

**Returns**

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Reimplemented from `std::basic_streambuf< _CharT, _Traits >`.

Definition at line 138 of file `stdio_sync_filebuf.h`.

## 4.942.2.36 underflow()

```
template<typename _CharT , typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf<_CharT, _Traits>::underflow () [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

## Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

## Note

Base class version does nothing, returns eof().

Reimplemented from `std::basic_streambuf<_CharT, _Traits>`.

Definition at line 131 of file `stdio_sync_filebuf.h`.

## 4.942.2.37 xsgetn()

```
template<typename _CharT , typename _Traits >
streamsize std::basic_streambuf<_CharT, _Traits>::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.

## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

**Returns**

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 46 of file `streambuf.tcc`.

**4.942.2.38 xsputn()**

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf<_CharT, _Traits>::xsputn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

**Parameters**

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

**Returns**

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Reimplemented in [std::basic\\_filebuf<\\_CharT, \\_Traits>](#), [std::basic\\_filebuf<\\_CharT, encoding\\_char\\_traits<\\_CharT>>](#), and [std::basic\\_filebuf<char\\_type, traits\\_type>](#).

Definition at line 80 of file `streambuf.tcc`.

**4.942.3 Member Data Documentation**

#### 4.942.3.1 \_M\_buf\_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf<_CharT, _Traits>::_M_buf_locale [protected], [inherited]
```

Current locale setting.

Definition at line 199 of file streambuf.

#### 4.942.3.2 \_M\_in\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_beg [protected], [inherited]
```

Start of get area.

Definition at line 191 of file streambuf.

#### 4.942.3.3 \_M\_in\_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_cur [protected], [inherited]
```

Current read area.

Definition at line 192 of file streambuf.

#### 4.942.3.4 \_M\_in\_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_in_end [protected], [inherited]
```

End of get area.

Definition at line 193 of file streambuf.

#### 4.942.3.5 \_M\_out\_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf<_CharT, _Traits>::_M_out_beg [protected], [inherited]
```

Start of put area.

Definition at line 194 of file streambuf.

#### 4.942.3.6 `_M_out_cur`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file streambuf.

#### 4.942.3.7 `_M_out_end`

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file streambuf.

The documentation for this class was generated from the following file:

- [stdio\\_sync\\_filebuf.h](#)

### 4.943 `std::chrono::_V2::steady_clock` Struct Reference

#### Public Types

- typedef [chrono::nanoseconds](#) **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef [chrono::time\\_point](#)< [steady\\_clock](#), [duration](#) > **time\_point**

#### Static Public Member Functions

- static [time\\_point](#) **now** () noexcept

#### Static Public Attributes

- static constexpr bool **is\_steady**

#### 4.943.1 Detailed Description

Monotonic clock.

Time returned has the property of only increasing at a uniform rate.

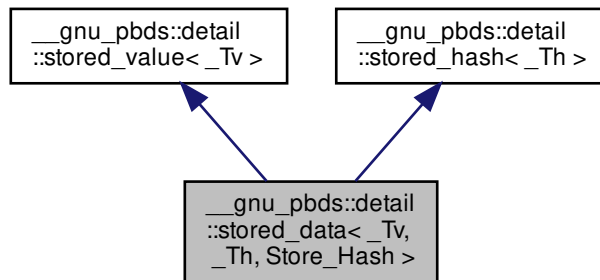
Definition at line 1078 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

4.944 `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >`:



## Public Types

- typedef `_Th` **hash\_type**
- typedef `_Tv` **value\_type**

## Public Attributes

- hash\_type **m\_hash**
- value\_type **m\_value**

## 4.944.1 Detailed Description

```
template<typename _Tv, typename _Th, bool Store_Hash>
struct __gnu_pbds::detail::stored_data<_Tv, _Th, Store_Hash >
```

Primary template for representation of stored data. Two types of data can be stored: value and hash.

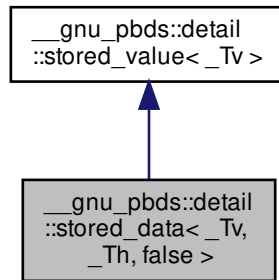
Definition at line 95 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.945 `__gnu_pbds::detail::stored_data<_Tv, _Th, false >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_data<_Tv, _Th, false >`:



##### Public Types

- `typedef _Tv value_type`

##### Public Attributes

- `value_type m_value`

##### 4.945.1 Detailed Description

```
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data<_Tv, _Th, false >
```

Specialization for representation of stored data of just value type.

Definition at line 101 of file `types_traits.hpp`.

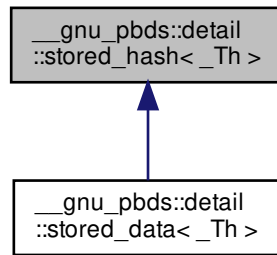
The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)



4.946 `__gnu_pbds::detail::stored_hash<_Th>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_hash<_Th>`:



#### Public Types

- typedef `_Th` **hash\_type**

#### Public Attributes

- `hash_type` **m\_hash**

#### 4.946.1 Detailed Description

```
template<typename _Th>
struct __gnu_pbds::detail::stored_hash<_Th>
```

Stored hash.

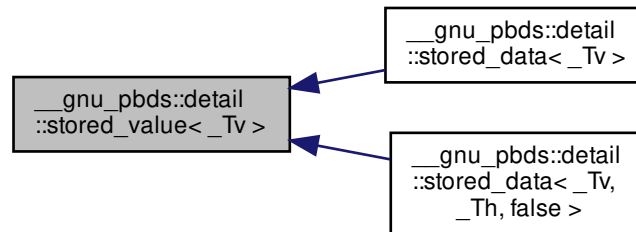
Definition at line 86 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.947 `__gnu_pbds::detail::stored_value<_Tv>` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::stored_value<_Tv>`:



##### Public Types

- typedef `_Tv` **value\_type**

##### Public Attributes

- value\_type **m\_value**

##### 4.947.1 Detailed Description

```
template<typename _Tv>
struct __gnu_pbds::detail::stored_value<_Tv>
```

Stored value.

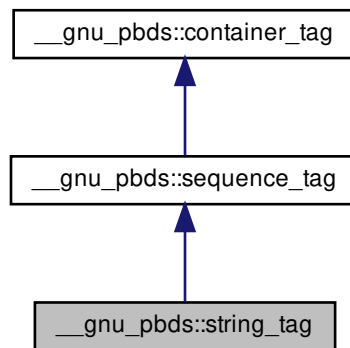
Definition at line 78 of file `types_traits.hpp`.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

## 4.948 \_\_gnu\_pbds::string\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::string\_tag:



### 4.948.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.

Definition at line 132 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

## 4.949 std::student\_t\_distribution<\_RealType> Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **student\_t\_distribution** (\_RealType \_\_n)
- **student\_t\_distribution** (const [param\\_type](#) & \_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator & \_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator & \_\_urng, const [param\\_type](#) & \_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator & \_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \* \_\_f, [result\\_type](#) \* \_\_t, \_UniformRandomNumberGenerator & \_\_urng, const [param\\_type](#) & \_\_p)
- [result\\_type](#) max () const
- [result\\_type](#) min () const
- \_RealType n () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator & \_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) operator() (\_UniformRandomNumberGenerator & \_\_urng, const [param\\_type](#) & \_\_p)
- [param\\_type](#) param () const
- void [param](#) (const [param\\_type](#) & \_\_param)
- void [reset](#) ()

## Friends

- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_ostream](#)< \_CharT, \_Traits > & [operator<<](#) ([std::basic\\_ostream](#)< \_CharT, \_Traits > & \_\_os, const [std::student\\_t\\_distribution](#)< \_RealType1 > & \_\_x)
- bool [operator==](#) (const [student\\_t\\_distribution](#) & \_\_d1, const [student\\_t\\_distribution](#) & \_\_d2)
- template<typename \_RealType1, typename \_CharT, typename \_Traits >  
[std::basic\\_istream](#)< \_CharT, \_Traits > & [operator>>](#) ([std::basic\\_istream](#)< \_CharT, \_Traits > & \_\_is, [std::student\\_t\\_distribution](#)< \_RealType1 > & \_\_x)

## 4.949.1 Detailed Description

```
template<typename _RealType = double>
class std::student_t_distribution< _RealType >
```

A [student\\_t\\_distribution](#) random number distribution.

The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} \left(1 + \frac{x^2}{n}\right)^{-(n+1)/2}$$

Definition at line 3294 of file random.h.

#### 4.949.2 Member Typedef Documentation

##### 4.949.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::student_t_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 3297 of file random.h.

#### 4.949.3 Member Function Documentation

##### 4.949.3.1 max()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 3386 of file random.h.

References std::numeric\_limits<\_Tp>::max().

##### 4.949.3.2 min()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 3379 of file random.h.

References std::numeric\_limits<\_Tp>::lowest().

##### 4.949.3.3 n()

```
template<typename _RealType = double>
_RealType std::student_t_distribution< _RealType >::n () const [inline]
```

Definition at line 3357 of file random.h.

Referenced by std::student\_t\_distribution<\_RealType>::operator()().

#### 4.949.3.4 operator()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::student_t_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 3394 of file random.h.

References `std::student_t_distribution< _RealType >::n()`, and `std::sqrt()`.

#### 4.949.3.5 param() [1/2]

```
template<typename _RealType = double>
param_type std::student_t_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 3364 of file random.h.

#### 4.949.3.6 param() [2/2]

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 3372 of file random.h.

#### 4.949.3.7 reset()

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 3347 of file random.h.

References `std::normal_distribution< _RealType >::reset()`, and `std::gamma_distribution< _RealType >::reset()`.

## 4.949.4 Friends And Related Function Documentation

## 4.949.4.1 operator&lt;&lt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream<_CharT, _Traits > & __os,
 const std::student_t_distribution<_RealType1 > & __x) [friend]
```

Inserts a student\_t\_distribution random number distribution \_\_x into the output stream \_\_os.

## Parameters

|      |                                                      |
|------|------------------------------------------------------|
| __os | An output stream.                                    |
| __x  | A student_t_distribution random number distribution. |

## Returns

The output stream with the state of \_\_x inserted or in an error state.

## 4.949.4.2 operator==

```
template<typename _RealType = double>
bool operator== (
 const student_t_distribution<_RealType > & __d1,
 const student_t_distribution<_RealType > & __d2) [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

Definition at line 3443 of file random.h.

## 4.949.4.3 operator&gt;&gt;

```
template<typename _RealType = double>
template<typename _RealType1 , typename _CharT , typename _Traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream<_CharT, _Traits > & __is,
 std::student_t_distribution<_RealType1 > & __x) [friend]
```

Extracts a student\_t\_distribution random number distribution \_\_x from the input stream \_\_is.

## Parameters

|                         |                                                                       |
|-------------------------|-----------------------------------------------------------------------|
| <code>_↔<br/>_is</code> | An input stream.                                                      |
| <code>_↔<br/>_x</code>  | A <code>student_t_distribution</code> random number generator engine. |

## Returns

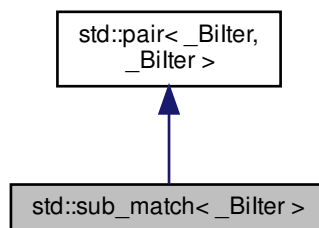
The input stream with `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

4.950 `std::sub_match<_Bilter >` Class Template Reference

Inheritance diagram for `std::sub_match<_Bilter >`:



## Public Types

- typedef `__iter_traits::difference_type` **difference\_type**
- typedef `_Bilter` **first\_type**
- typedef `_Bilter` **iterator**
- typedef `_Bilter` **second\_type**
- typedef `basic_string<value_type>` **string\_type**
- typedef `__iter_traits::value_type` **value\_type**



## Public Member Functions

- int [compare](#) (const [sub\\_match](#) &\_\_s) const
- difference\_type [length](#) () const noexcept
- [operator string\\_type](#) () const
- [string\\_type](#) [str](#) () const
- constexpr void [swap](#) ([pair](#) &\_\_p) noexcept(\_\_and\_< \_\_is\_nothrow\_swappable< \_Bilter >, \_\_is\_nothrow\_swappable< \_Bilter >>::value)
- int [compare](#) (const [string\\_type](#) &\_\_s) const
- int [compare](#) (const value\_type \*\_\_s) const

## Public Attributes

- \_Bilter [first](#)
- bool **matched**
- \_Bilter [second](#)

## Related Functions

(Note that these are not member functions.)

- constexpr [pair](#)< typename \_\_decay\_and\_strip< \_Bilter >::\_\_type, typename \_\_decay\_and\_strip< \_Bilter >::\_\_type > [make\\_pair](#) (\_Bilter &&\_\_x, \_Bilter &&\_\_y)
- template<typename \_Bilter >  
bool [operator==](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bilter >  
bool [operator!=](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bilter >  
bool [operator<](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bilter >  
bool [operator<=](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bilter >  
bool [operator>=](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bilter >  
bool [operator>](#) (const [sub\\_match](#)< \_Bilter > &\_\_lhs, const [sub\\_match](#)< \_Bilter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc >  
bool [operator==](#) (const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const [sub\\_match](#)< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter, typename \_Ch\_traits, typename \_Ch\_alloc >  
bool [operator!=](#) (const \_\_sub\_match\_string< \_Bi\_iter, \_Ch\_traits, \_Ch\_alloc > &\_\_lhs, const [sub\\_match](#)< \_Bi\_iter > &\_\_rhs)



- template<typename \_Bi\_iter >  
bool operator< (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator> (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator>= (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator<= (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator== (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator!= (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator< (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator> (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator>= (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator<= (typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_lhs, const sub\_match< \_Bi\_iter > &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator== (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator!= (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator< (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator> (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator>= (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Bi\_iter >  
bool operator<= (const sub\_match< \_Bi\_iter > &\_\_lhs, typename iterator\_traits< \_Bi\_iter >::value\_type const &\_\_rhs)
- template<typename \_Ch\_type, typename \_Ch\_traits, typename \_Bi\_iter >  
basic\_ostream< \_Ch\_type, \_Ch\_traits > & operator<< (basic\_ostream< \_Ch\_type, \_Ch\_traits > &\_\_os, const sub\_match< \_Bi\_iter > &\_\_m)

- constexpr `enable_if< __and< __is_swappable< _Bilter >, __is_swappable< _Bilter > >::value >::type swap(pair< _Bilter, _Bilter > &__x, pair< _Bilter, _Bilter > &__y) noexcept(noexcept(__x.swap(__y)))`
- constexpr bool `operator==` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)
- constexpr bool `operator<` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)
- constexpr bool `operator!=` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)
- constexpr bool `operator>` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)
- constexpr bool `operator<=` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)
- constexpr bool `operator>=` (const pair< \_Bilter, \_Bilter > &\_\_x, const pair< \_Bilter, \_Bilter > &\_\_y)

#### 4.950.1 Detailed Description

```
template<typename _Bilter>
class std::sub_match< _Bilter >
```

A sequence of characters matched by a particular marked sub-expression.

An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with `std::basic_string` objects of a similar base character type as the pattern matched by the regular expression.

The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Definition at line 868 of file `regex.h`.

#### 4.950.2 Member Typedef Documentation

##### 4.950.2.1 first\_type

```
typedef _BiIter std::pair< _BiIter , _BiIter >::first_type [inherited]
```

The type of the `first` member.

Definition at line 214 of file `stl_pair.h`.

##### 4.950.2.2 second\_type

```
typedef _BiIter std::pair< _BiIter , _BiIter >::second_type [inherited]
```

The type of the `second` member.

Definition at line 215 of file `stl_pair.h`.

#### 4.950.3 Member Function Documentation

##### 4.950.3.1 compare() [1/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
 const sub_match< _BiIter > &__s) const [inline]
```

Compares this and another matched sequence.

## Parameters

|       |                                                  |
|-------|--------------------------------------------------|
| $\_s$ | Another matched sequence to compare to this one. |
|-------|--------------------------------------------------|

## Return values

|      |                                                   |
|------|---------------------------------------------------|
| $<0$ | this matched sequence will collate before $\_s$ . |
| $=0$ | this matched sequence is equivalent to $\_s$ .    |
| $>0$ | this matched sequence will collate after $\_s$ .  |

Definition at line 923 of file regex.h.

Referenced by `std::sub_match<_Bi_iter>::operator!=()`, `std::sub_match<_Bi_iter>::operator<()`, `std::sub_match<_Bi_iter>::operator<=()`, `std::sub_match<_Bi_iter>::operator==()`, `std::sub_match<_Bi_iter>::operator>()`, and `std::sub_match<_Bi_iter>::operator>=()`.

## 4.950.3.2 compare() [2/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter>::compare (
 const string_type & __s) const [inline]
```

Compares this sub\_match to a string.

## Parameters

|       |                                        |
|-------|----------------------------------------|
| $\_s$ | A string to compare to this sub_match. |
|-------|----------------------------------------|

## Return values

|      |                                                   |
|------|---------------------------------------------------|
| $<0$ | this matched sequence will collate before $\_s$ . |
| $=0$ | this matched sequence is equivalent to $\_s$ .    |
| $>0$ | this matched sequence will collate after $\_s$ .  |

Definition at line 937 of file regex.h.

## 4.950.3.3 compare() [3/3]

```
template<typename _BiIter>
int std::sub_match<_BiIter>::compare (
 const value_type * __s) const [inline]
```

Compares this sub\_match to a string.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__s</code> | A string to compare to this <code>sub_match</code> . |
|------------------|------------------------------------------------------|

**Return values**

|                     |                                                              |
|---------------------|--------------------------------------------------------------|
| <code>&lt; 0</code> | this matched sequence will collate before <code>__s</code> . |
| <code>= 0</code>    | this matched sequence is equivalent to <code>__s</code> .    |
| <code>&gt; 0</code> | this matched sequence will collate after <code>__s</code> .  |

Definition at line 941 of file `regex.h`.

**4.950.3.4 `length()`**

```
template<typename _BiIter>
difference_type std::sub_match< _BiIter >::length () const [inline], [noexcept]
```

Gets the length of the matching sequence.

Definition at line 884 of file `regex.h`.

**4.950.3.5 `operator string_type()`**

```
template<typename _BiIter>
std::sub_match< _BiIter >::operator string_type () const [inline]
```

Gets the matching sequence as a string.

**Returns**

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the `str()` member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

Definition at line 897 of file `regex.h`.

## 4.950.3.6 str()

```
template<typename _BiIter>
string_type std::sub_match<_BiIter >::str () const [inline]
```

Gets the matching sequence as a string.

## Returns

the matching sequence as a string.

Definition at line 906 of file regex.h.

Referenced by std::sub\_match<\_Bi\_iter >::operator string\_type(), and std::sub\_match<\_Bi\_iter >::operator<<().

## 4.950.3.7 swap()

```
constexpr void std::pair<_BiIter , _BiIter >::swap (
 pair<_BiIter, _BiIter > & __p) [inline], [noexcept], [inherited]
```

Swap the first members and then the second members.

Definition at line 439 of file stl\_pair.h.

## 4.950.4 Friends And Related Function Documentation

## 4.950.4.1 make\_pair()

```
constexpr pair< typename __decay_and_strip<_BiIter >::__type, typename __decay_and_strip<_BiIter >::__type > make_pair (
 _BiIter && __x,
 _BiIter && __y) [related]
```

A convenience wrapper for creating a pair from two objects.

## Parameters

|       |                    |
|-------|--------------------|
| $\_x$ | The first object.  |
| $\_y$ | The second object. |

**Returns**

A newly-constructed `pair<>` object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).

Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.

Definition at line 567 of file `stl_pair.h`.

**4.950.4.2 `operator!=()`**

```
constexpr bool operator!= (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Uses `operator==` to find the result.

Definition at line 496 of file `stl_pair.h`.

**4.950.4.3 `operator<()`**

```
constexpr bool operator< (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of the same type, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

Definition at line 489 of file `stl_pair.h`.

**4.950.4.4 `operator<=()`**

```
constexpr bool operator<= (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Uses `operator<` to find the result.

Definition at line 508 of file `stl_pair.h`.



#### 4.950.4.5 operator==()

```
constexpr bool operator== (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Two pairs of the same type are equal iff their members are equal.

Definition at line 466 of file stl\_pair.h.

#### 4.950.4.6 operator>()

```
constexpr bool operator> (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Uses operator< to find the result.

Definition at line 502 of file stl\_pair.h.

#### 4.950.4.7 operator>=()

```
constexpr bool operator>= (
 const pair< _BiIter , _BiIter > & __x,
 const pair< _BiIter , _BiIter > & __y) [related]
```

Uses operator< to find the result.

Definition at line 514 of file stl\_pair.h.

#### 4.950.4.8 swap()

```
constexpr enable_if< __and< __is_swappable< _BiIter >, __is_swappable< _BiIter > >::value >↔
::type swap (
 pair< _BiIter , _BiIter > & __x,
 pair< _BiIter , _BiIter > & __y) [related]
```

Swap overload for pairs. Calls std::pair::swap().

#### Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see <https://gcc.gnu.org/PR38466>

Definition at line 533 of file stl\_pair.h.

#### 4.950.5 Member Data Documentation

##### 4.950.5.1 first

`_BiIter` [std::pair](#)< `_BiIter` , `_BiIter` >::first [inherited]

The first member.

Definition at line 217 of file `stl_pair.h`.

##### 4.950.5.2 second

`_BiIter` [std::pair](#)< `_BiIter` , `_BiIter` >::second [inherited]

The second member.

Definition at line 218 of file `stl_pair.h`.

The documentation for this class was generated from the following file:

- [regex.h](#)

#### 4.951 `std::subtract_with_carry_engine<_UIntType, __w, __s, __r>` Class Template Reference

##### Public Types

- typedef `_UIntType` [result\\_type](#)

##### Public Member Functions

- [subtract\\_with\\_carry\\_engine](#) ([result\\_type](#) \_\_sd)
- template<typename `_Sseq` , typename = `_If_seed_seq<_Sseq>`>  
[subtract\\_with\\_carry\\_engine](#) (`_Sseq` &\_\_q)
- void [discard](#) (unsigned long long \_\_z)
- [result\\_type](#) operator() ()
- template<typename `_Sseq` >  
auto [seed](#) (`_Sseq` &\_\_q) -> `_If_seed_seq<_Sseq>`
- void [seed](#) ([result\\_type](#) \_\_sd=default\_seed)
- template<typename `_Sseq` >  
`_If_seed_seq<_Sseq>` [seed](#) (`_Sseq` &\_\_q)

## Static Public Member Functions

- static constexpr [result\\_type](#) max ()
- static constexpr [result\\_type](#) min ()

## Static Public Attributes

- static constexpr [result\\_type](#) default\_seed
- static constexpr size\_t long\_lag
- static constexpr size\_t short\_lag
- static constexpr size\_t word\_size

## Friends

- template<typename \_UIntType1 , size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT , typename \_Traits >  
std::basic\_ostream< \_CharT, \_Traits > & operator<< (std::basic\_ostream< \_CharT, \_Traits > &\_\_os, const  
std::subtract\_with\_carry\_engine< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)
- bool operator== (const subtract\_with\_carry\_engine &\_\_lhs, const subtract\_with\_carry\_engine &\_\_rhs)
- template<typename \_UIntType1 , size\_t \_\_w1, size\_t \_\_s1, size\_t \_\_r1, typename \_CharT , typename \_Traits >  
std::basic\_istream< \_CharT, \_Traits > & operator>> (std::basic\_istream< \_CharT, \_Traits > &\_\_is,  
std::subtract\_with\_carry\_engine< \_UIntType1, \_\_w1, \_\_s1, \_\_r1 > &\_\_x)

## 4.951.1 Detailed Description

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >
```

The Marsaglia-Zaman generator.

This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.

A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is  $r$  and the maximum period of the generator is  $(m^r - m^s - 1)$ .

Definition at line 692 of file random.h.

## 4.951.2 Member Typedef Documentation

#### 4.951.2.1 result\_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

Definition at line 707 of file random.h.

#### 4.951.3 Constructor & Destructor Documentation

##### 4.951.3.1 subtract\_with\_carry\_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
 result_type __sd) [inline], [explicit]
```

Constructs an explicitly seeded subtract\_with\_carry\_engine random number generator.

Definition at line 723 of file random.h.

References `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed()`.

##### 4.951.3.2 subtract\_with\_carry\_engine() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq , typename = _If_seed_seq<_Sseq>>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
 _Sseq & __q) [inline], [explicit]
```

Constructs a subtract\_with\_carry\_engine random number engine seeded from the seed sequence `__q`.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__q</code> | the seed sequence. |
|------------------|--------------------|

Definition at line 734 of file random.h.

References `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed()`.

#### 4.951.4 Member Function Documentation

#### 4.951.4.1 discard()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::discard (
 unsigned long long __z) [inline]
```

Discard a sequence of random numbers.

Definition at line 780 of file random.h.

#### 4.951.4.2 max()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::max ()
[inline], [static]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

Definition at line 773 of file random.h.

#### 4.951.4.3 min()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::min ()
[inline], [static]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

Definition at line 765 of file random.h.

#### 4.951.4.4 operator()()

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type std::subtract_with_carry_engine<
_UIntType, __w, __s, __r >::operator() ()
```

Gets the next random number in the sequence.

Definition at line 594 of file bits/random.tcc.

## 4.951.4.5 seed() [1/2]

```
template<typename _UIntType , size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
 result_type __sd = default_seed)
```

Seeds the initial state  $x_0$  of the random number generator.

N1688[4.19] modifies this as follows. If `__value == 0`, sets value to 19780503. In any case, with a linear congruential generator `lcg(i)` having parameters  $m_{lcg} = 2147483563$ ,  $a_{lcg} = 40014$ ,  $c_{lcg} = 0$ , and  $lcg(0) = value$ , sets  $x_r \dots x_{-1}$  to  $lcg(1) \bmod m \dots lcg(r) \bmod m$  respectively. If  $x_{-1} = 0$  set carry to 1, otherwise sets carry to 0.

Definition at line 538 of file `bits/random.tcc`.

Referenced by `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine()`.

## 4.951.4.6 seed() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq >
_If_seed_seq<_Sseq> std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
 _Sseq & __q)
```

Seeds the initial state  $x_0$  of the % `subtract_with_carry_engine` random number generator.

## 4.951.5 Friends And Related Function Documentation

## 4.951.5.1 operator&lt;&lt;

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1 , size_t __w1, size_t __s1, size_t __r1, typename _CharT , typename
_Traits >
std::basic_ostream<_CharT, _Traits>& operator<< (
 std::basic_ostream< _CharT, _Traits > & __os,
 const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]
```

Inserts the current state of a % `subtract_with_carry_engine` random number generator engine `__x` into the output stream `__os`.

## Parameters

|                   |                                                                             |
|-------------------|-----------------------------------------------------------------------------|
| <code>__os</code> | An output stream.                                                           |
| <code>__x</code>  | A % <code>subtract_with_carry_engine</code> random number generator engine. |

**Returns**

The output stream with the state of \_\_x inserted or in an error state.

**4.951.5.2 operator==**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
 const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
 const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs) [friend]
```

Compares two % subtract\_with\_carry\_engine random number generator objects of the same type for equality.

**Parameters**

|                    |                                                                      |
|--------------------|----------------------------------------------------------------------|
| <code>__lhs</code> | A % subtract_with_carry_engine random number generator object.       |
| <code>__rhs</code> | Another % subtract_with_carry_engine random number generator object. |

**Returns**

true if the infinite sequences of generated values would be equal, false otherwise.

Definition at line 805 of file random.h.

**4.951.5.3 operator>>**

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename
_traits >
std::basic_istream<_CharT, _Traits>& operator>> (
 std::basic_istream< _CharT, _Traits > & __is,
 std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x) [friend]
```

Extracts the current state of a % subtract\_with\_carry\_engine random number generator engine \_\_x from the input stream \_\_is.

**Parameters**

|                   |                                                                |
|-------------------|----------------------------------------------------------------|
| <code>__is</code> | An input stream.                                               |
| <code>__x</code>  | A % subtract_with_carry_engine random number generator engine. |

**Returns**

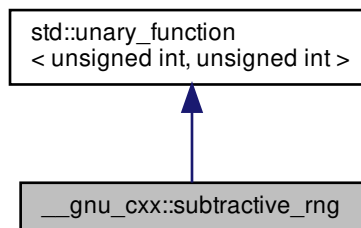
The input stream with the state of `__x` extracted or in an error state.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

**4.952 `__gnu_cxx::subtractive_rng` Class Reference**

Inheritance diagram for `__gnu_cxx::subtractive_rng`:

**Public Types**

- typedef unsigned int [argument\\_type](#)
- typedef unsigned int [result\\_type](#)

**Public Member Functions**

- [subtractive\\_rng](#) (unsigned int `__seed`)
- [subtractive\\_rng](#) ()
- void **`_M_initialize`** (unsigned int `__seed`)
- unsigned int [operator\(\)](#) (unsigned int `__limit`)

**4.952.1 Detailed Description**

The `subtractive_rng` class is documented on [SGI's site](#). Note that this code assumes that `int` is 32 bits.

Definition at line 344 of file `ext/functional`.



#### 4.952.2 Member Typedef Documentation

##### 4.952.2.1 `argument_type`

```
typedef unsigned int std::unary_function< unsigned int , unsigned int >::argument_type [inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

##### 4.952.2.2 `result_type`

```
typedef unsigned int std::unary_function< unsigned int , unsigned int >::result_type [inherited]
```

`result_type` is the return type

Definition at line 111 of file `stl_function.h`.

#### 4.952.3 Constructor & Destructor Documentation

##### 4.952.3.1 `subtractive_rng()` [1/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng (
 unsigned int __seed) [inline]
```

Ctor allowing you to initialize the seed.

Definition at line 386 of file `ext/functional`.

##### 4.952.3.2 `subtractive_rng()` [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng () [inline]
```

Default ctor; initializes its state with some number you don't see.

Definition at line 390 of file `ext/functional`.

#### 4.952.4 Member Function Documentation

#### 4.952.4.1 operator()

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
 unsigned int __limit) [inline]
```

Returns a number less than the argument.

Definition at line 355 of file ext/functional.

The documentation for this class was generated from the following file:

- [ext/functional](#)

#### 4.953 \_\_gnu\_pbds::detail::synth\_access\_traits< Type\_Traits, Set, \_ATraits > Struct Template Reference

Inherits `_ATraits`.

##### Public Types

- typedef `_ATraits` **base\_type**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `type_traits::const_reference` **const\_reference**
- typedef `type_traits::key_const_reference` **key\_const\_reference**
- typedef `Type_Traits` **type\_traits**

##### Public Member Functions

- **synth\_access\_traits** (const `base_type` &)
- bool **cmp\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **cmp\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=false) const
- bool **equal\_keys** (key\_const\_reference, key\_const\_reference) const
- bool **equal\_prefixes** (const\_iterator, const\_iterator, const\_iterator, const\_iterator, bool compare\_after=true) const

##### Static Public Member Functions

- static key\_const\_reference **extract\_key** (const\_reference)

#### 4.953.1 Detailed Description

```
template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >
```

Synthetic element access traits.

Definition at line 59 of file synth\_access\_traits.hpp.

The documentation for this struct was generated from the following file:

- [synth\\_access\\_traits.hpp](#)

## 4.954 `std::chrono::_V2::system_clock` Struct Reference

### Public Types

- typedef `chrono::nanoseconds` **duration**
- typedef `duration::period` **period**
- typedef `duration::rep` **rep**
- typedef `chrono::time_point< system_clock, duration >` **time\_point**

### Static Public Member Functions

- static `time_point from_time_t` (`std::time_t __t`) noexcept
- static `time_point now` () noexcept
- static `std::time_t to_time_t` (const `time_point &__t`) noexcept

### Static Public Attributes

- static constexpr bool **is\_steady**

#### 4.954.1 Detailed Description

System clock.

Time returned represents wall time from the system-wide clock.

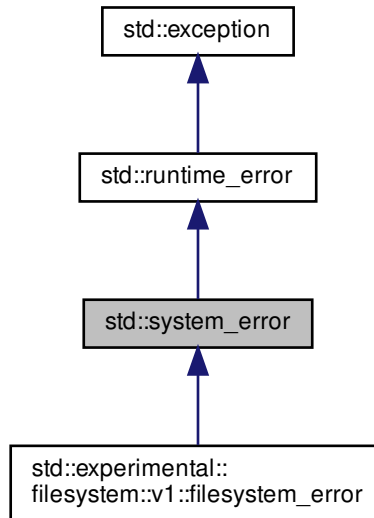
Definition at line 1038 of file `chrono`.

The documentation for this struct was generated from the following file:

- `chrono`

## 4.955 std::system\_error Class Reference

Inheritance diagram for std::system\_error:



### Public Member Functions

- **system\_error** ([error\\_code](#) \_\_ec=[error\\_code](#)())
- **system\_error** ([error\\_code](#) \_\_ec, const [string](#) &\_\_what)
- **system\_error** ([error\\_code](#) \_\_ec, const char \* \_\_what)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const char \* \_\_what)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat)
- **system\_error** (int \_\_v, const error\_category &\_\_ecat, const [string](#) &\_\_what)
- **system\_error** (const [system\\_error](#) &)=default
- const [error\\_code](#) & **code** () const noexcept
- [system\\_error](#) & **operator=** (const [system\\_error](#) &)=default
- virtual const char \* **what** () const noexcept

### 4.955.1 Detailed Description

An exception type that includes an `error_code` value.

Typically used to report errors from the operating system and other low-level APIs.

Definition at line 428 of file `system_error`.

## 4.955.2 Member Function Documentation

4.955.2.1 `what()`

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

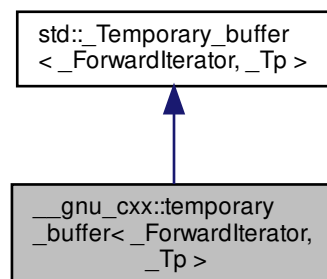
Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [system\\_error](#)

4.956 `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::temporary_buffer<_ForwardIterator, _Tp>`:



## Public Types

- typedef pointer **iterator**
- typedef value\_type \* **pointer**
- typedef ptrdiff\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- [temporary\\_buffer](#) ([\\_ForwardIterator](#) \_\_first, [\\_ForwardIterator](#) \_\_last)
- [~temporary\\_buffer](#) ()
- iterator [begin](#) ()
- iterator [end](#) ()
- [size\\_type](#) [requested\\_size](#) () const
- [size\\_type](#) [size](#) () const

## Protected Attributes

- pointer [\\_M\\_buffer](#)
- [size\\_type](#) [\\_M\\_len](#)
- [size\\_type](#) [\\_M\\_original\\_len](#)

### 4.956.1 Detailed Description

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >
```

This class provides similar behavior and semantics of the standard functions `get_temporary_buffer()` and `return_temporary_buffer()`, but encapsulated in a type vaguely resembling a standard container.

By default, a `temporary_buffer<Iter>` stores space for objects of whatever type the `Iter` iterator points to. It is constructed from a typical `[first,last)` range, and provides the `begin()`, `end()`, `size()` functions, as well as `requested_size()`. For non-trivial types, copies of `*first` will be used to initialize the storage.

`malloc` is used to obtain underlying storage.

Like `get_temporary_buffer()`, not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of `[first,last)`, but if `size()` is less than `requested_size()`, then this didn't happen.

Definition at line 184 of file `ext/memory`.

### 4.956.2 Constructor & Destructor Documentation

#### 4.956.2.1 temporary\_buffer()

```
template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (
 _ForwardIterator __first,
 _ForwardIterator __last) [inline]
```

Requests storage large enough to hold a copy of `[first,last)`.

Definition at line 187 of file `ext/memory`.

#### 4.956.2.2 ~temporary\_buffer()

```
template<class _ForwardIterator , class _Tp = typename std::iterator_traits<_ForwardIterator>←
::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::~~temporary_buffer () [inline]
```

Destroys objects and frees storage.

Definition at line 193 of file ext/memory.

### 4.956.3 Member Function Documentation

#### 4.956.3.1 begin()

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin () [inline], [inherited]
```

As per Table mumble.

Definition at line 165 of file stl\_tempbuf.h.

#### 4.956.3.2 end()

```
template<typename _ForwardIterator , typename _Tp >
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end () [inline], [inherited]
```

As per Table mumble.

Definition at line 170 of file stl\_tempbuf.h.

#### 4.956.3.3 requested\_size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::requested_size () const [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

Definition at line 160 of file stl\_tempbuf.h.

## 4.956.3.4 size()

```
template<typename _ForwardIterator , typename _Tp >
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const [inline], [inherited]
```

As per Table mumble.

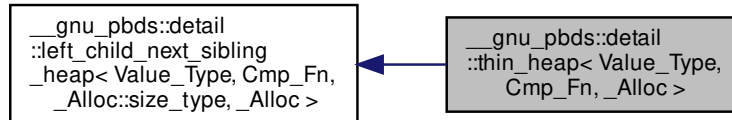
Definition at line 155 of file stl\_tempbuf.h.

The documentation for this struct was generated from the following file:

- [ext/memory](#)

## 4.957 \_\_gnu\_pbds::detail::thin\_heap&lt; Value\_Type, Cmp\_Fn, \_Alloc &gt; Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::thin\_heap< Value\_Type, Cmp\_Fn, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `base_type::const_iterator` **const\_iterator**
- typedef `__rebind_a::const_pointer` **const\_pointer**
- typedef `__rebind_a::const_reference` **const\_reference**
- typedef `_Alloc::difference_type` **difference\_type**
- typedef `base_type::iterator` **iterator**
- typedef `base_type::point_const_iterator` **point\_const\_iterator**
- typedef `base_type::point_iterator` **point\_iterator**
- typedef `__rebind_a::pointer` **pointer**
- typedef `__rebind_a::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `Value_Type` **value\_type**



## Public Member Functions

- `iterator begin ()`
- `const_iterator begin () const`
- `void clear ()`
- `bool empty () const`
- `iterator end ()`
- `const_iterator end () const`
- `void erase (point_iterator)`
- `template<typename Pred >`  
`size_type erase_if (Pred)`
- `Cmp_Fn & get_cmp_fn ()`
- `const Cmp_Fn & get_cmp_fn () const`
- `void join (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)`
- `size_type max_size () const`
- `void modify (point_iterator, const_reference)`
- `void pop ()`
- `point_iterator push (const_reference)`
- `size_type size () const`
- `template<typename Pred >`  
`void split (Pred, thin_heap< Value_Type, Cmp_Fn, _Alloc > &)`
- `void swap (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)`
- `const_reference top () const`

## Protected Types

- `typedef base_type::node node`
- `typedef alloc_traits::allocator_type node_allocator`
- `typedef base_type::node_const_pointer node_const_pointer`
- `typedef _Alloc::size_type node_metadata`
- `typedef base_type::node_pointer node_pointer`
- `typedef std::pair< node_pointer, node_pointer > node_pointer_pair`

## Protected Member Functions

- `thin_heap (const Cmp_Fn &)`
- `thin_heap (const thin_heap< Value_Type, Cmp_Fn, _Alloc > &)`
- `void actual_erase_node (node_pointer)`
- `void bubble_to_top (node_pointer)`
- `void clear_imp (node_pointer)`
- `template<typename It >`  
`void copy_from_range (It, It)`
- `node_pointer get_new_node_for_insert (const_reference)`
- `node_pointer prune (Pred)`
- `void swap (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)`
- `void swap_with_parent (node_pointer, node_pointer)`
- `void to_linked_list ()`
- `void value_swap (left_child_next_sibling_heap &)`

### Static Protected Member Functions

- static node\_pointer **parent** (node\_pointer)

### Protected Attributes

- node\_pointer **m\_p\_root**
- size\_type **m\_size**

#### 4.957.1 Detailed Description

```
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >
```

Thin heap.

See Tarjan and Kaplan.

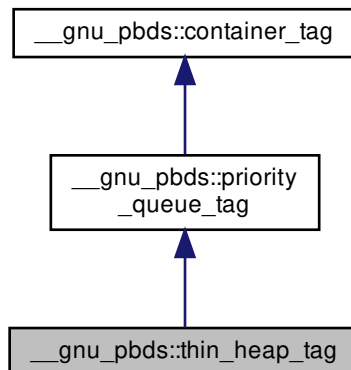
Definition at line 77 of file thin\_heap\_.hpp.

The documentation for this class was generated from the following file:

- [thin\\_heap\\_.hpp](#)

#### 4.958 \_\_gnu\_pbds::thin\_heap\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::thin\_heap\_tag:



## 4.958.1 Detailed Description

Thin heap.

Definition at line 186 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

4.959 `std::thread` Class Reference

## Classes

- class [id](#)

## Public Types

- using `_State_ptr` = [unique\\_ptr](#)< `_State` >
- typedef `__gthread_t` `native_handle_type`

## Public Member Functions

- `template<typename _Callable, typename... _Args, typename = _Require<__not_same<_Callable>>>`  
`thread` (`_Callable` &&\_\_f, `_Args` &&... \_\_args)
- `thread` (const [thread](#) &)=delete
- `thread` ([thread](#) &&\_\_t) noexcept
- void `detach` ()
- [id](#) `get_id` () const noexcept
- void `join` ()
- bool `joinable` () const noexcept
- `native_handle_type` `native_handle` ()
- [thread](#) & `operator=` (const [thread](#) &)=delete
- [thread](#) & `operator=` ([thread](#) &&\_\_t) noexcept
- void `swap` ([thread](#) &\_\_t) noexcept

## Static Public Member Functions

- `template<typename _Callable, typename... _Args>`  
static `_Invoker`< [\\_\\_decayed\\_tuple](#)< `_Callable`, `_Args...` > > `__make_invoker` (`_Callable` &&\_\_callable, `_Args` &&... \_\_args)
- static unsigned int `hardware_concurrency` () noexcept

#### 4.959.1 Detailed Description

thread

Definition at line 73 of file thread.

#### 4.959.2 Member Function Documentation

##### 4.959.2.1 native\_handle()

```
native_handle_type std::thread::native_handle () [inline]
```

##### Precondition

thread is joinable

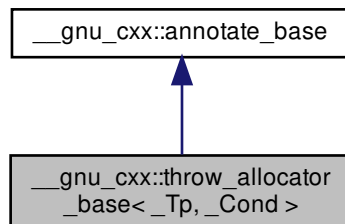
Definition at line 196 of file thread.

The documentation for this class was generated from the following file:

- [thread](#)

#### 4.960 \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::throw\_allocator\_base< \_Tp, \_Cond >:



## Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

## Public Member Functions

- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*hint=0)
- void **check** (size\_t label)
- void **check** (size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)
- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- template<typename \_Up, typename... \_Args>  
void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- template<typename \_Up >  
void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept

## Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)

## 4.960.1 Detailed Description

```
template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base<_Tp, _Cond>
```

Allocator class with logging and exception generation control. Intended to be used as an allocator\_type in templated code.

Note: Deallocate not allowed to throw.

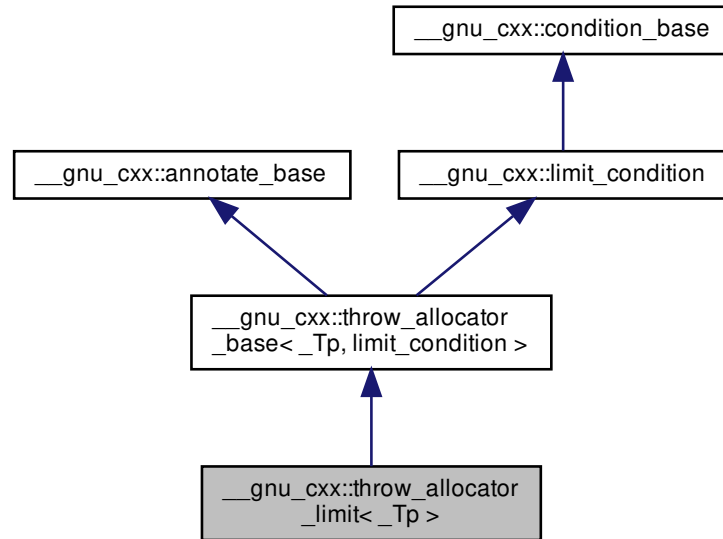
Definition at line 811 of file throw\_allocator.h.

The documentation for this class was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.961 `__gnu_cxx::throw_allocator_limit<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_limit<_Tp>`:



#### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#) &) noexcept
- template<typename \_Tp1 >  
**throw\_allocator\_limit** (const [throw\\_allocator\\_limit](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*hint=0)
- void **check** (size\_t label)
- void **check** (size\_type \_\_n)

- `map_alloc_type::iterator` **check\_allocated** (`void *p`, `size_t size`)
- `void` **check\_allocated** (`pointer __p`, `size_type __n`)
- `map_construct_type::iterator` **check\_constructed** (`void *p`)
- `void` **check\_constructed** (`size_t label`)
- `void` **construct** (`_Up *__p`, `_Args &&... __args`)
- `void` **deallocate** (`pointer __p`, `size_type __n`)
- `void` **destroy** (`_Up *__p`)
- `void` **erase** (`void *p`, `size_t size`)
- `void` **erase\_construct** (`void *p`)
- `void` **insert** (`void *p`, `size_t size`)
- `void` **insert\_construct** (`void *p`)
- `size_type` **max\_size** () `const noexcept`

#### Static Public Member Functions

- `static void` **check** ()
- `static size_t &` **count** ()
- `static size_t` **get\_label** ()
- `static size_t &` **limit** ()
- `static void` **set\_label** (`size_t l`)
- `static void` **set\_limit** (`const size_t __l`)
- `static void` **throw\_conditionally** ()

#### 4.961.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit<_Tp>
```

Allocator throwing via limit condition.

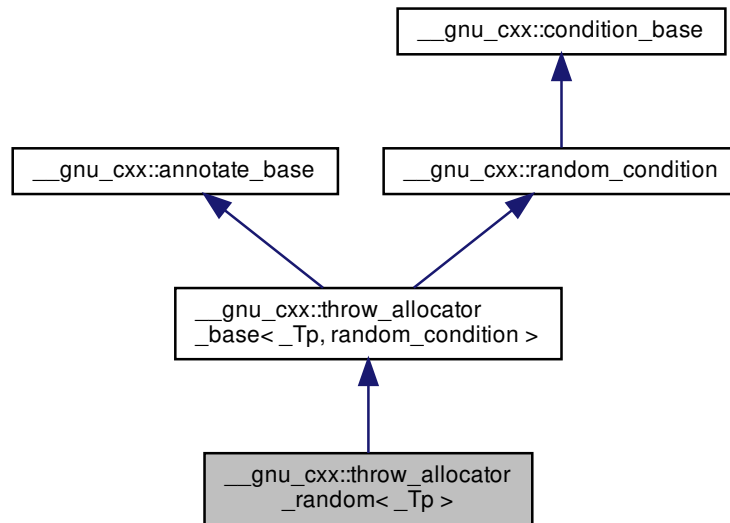
Definition at line 924 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.962 `__gnu_cxx::throw_allocator_random<_Tp>` Struct Template Reference

Inheritance diagram for `__gnu_cxx::throw_allocator_random<_Tp>`:



#### Public Types

- typedef const value\_type \* **const\_pointer**
- typedef const value\_type & **const\_reference**
- typedef std::ptrdiff\_t **difference\_type**
- typedef value\_type \* **pointer**
- typedef [std::true\\_type](#) **propagate\_on\_container\_move\_assignment**
- typedef value\_type & **reference**
- typedef std::size\_t **size\_type**
- typedef \_Tp **value\_type**

#### Public Member Functions

- **throw\_allocator\_random** (const [throw\\_allocator\\_random](#) &) noexcept
- template<typename \_Tp1 >  
  **throw\_allocator\_random** (const [throw\\_allocator\\_random](#)<\_Tp1 > &) noexcept
- pointer **address** (reference \_\_x) const noexcept
- const\_pointer **address** (const\_reference \_\_x) const noexcept
- pointer **allocate** (size\_type \_\_n, const void \*hint=0)
- void **check** (size\_t label)
- void **check** (size\_type \_\_n)
- map\_alloc\_type::iterator **check\_allocated** (void \*p, size\_t size)



- void **check\_allocated** (pointer \_\_p, size\_type \_\_n)
- map\_construct\_type::iterator **check\_constructed** (void \*p)
- void **check\_constructed** (size\_t label)
- void **construct** (\_Up \*\_\_p, \_Args &&... \_\_args)
- void **deallocate** (pointer \_\_p, size\_type \_\_n)
- void **destroy** (\_Up \*\_\_p)
- void **erase** (void \*p, size\_t size)
- void **erase\_construct** (void \*p)
- void **insert** (void \*p, size\_t size)
- void **insert\_construct** (void \*p)
- size\_type **max\_size** () const noexcept
- void **seed** (unsigned long \_\_s)

#### Static Public Member Functions

- static void **check** ()
- static size\_t **get\_label** ()
- static void **set\_label** (size\_t l)
- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

#### 4.962.1 Detailed Description

```
template<typename _Tp>
struct __gnu_cxx::throw_allocator_random<_Tp>
```

Allocator throwing via random condition.

Definition at line 946 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.963 `__gnu_cxx::throw_value_base<_Cond>` Struct Template Reference

Inherits `_Cond`.

#### Public Types

- typedef `_Cond` **condition\_type**

#### Public Member Functions

- **throw\_value\_base** (const [throw\\_value\\_base](#) &\_\_v)
- **throw\_value\_base** ([throw\\_value\\_base](#) &&)=default
- **throw\_value\_base** (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_base](#) & **operator=** (const [throw\\_value\\_base](#) &\_\_v)
- [throw\\_value\\_base](#) & **operator=** ([throw\\_value\\_base](#) &&)=default

#### Public Attributes

- std::size\_t **M\_i**

#### 4.963.1 Detailed Description

```
template<typename _Cond>
struct __gnu_cxx::throw_value_base< _Cond >
```

Class with exception generation control. Intended to be used as a value\_type in templated code.

Note: Destructor not allowed to throw.

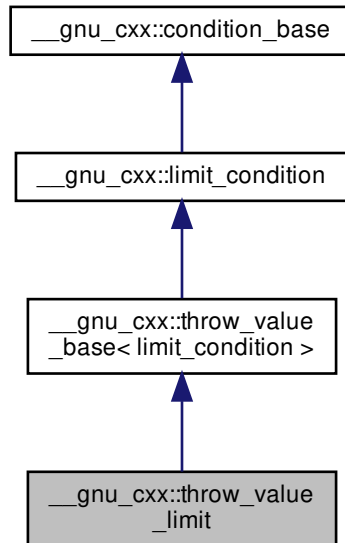
Definition at line 623 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.964 \_\_gnu\_cxx::throw\_value\_limit Struct Reference

Inheritance diagram for \_\_gnu\_cxx::throw\_value\_limit:



## Public Types

- typedef [throw\\_value\\_base< limit\\_condition >](#) **base\_type**
- typedef [limit\\_condition](#) **condition\_type**

## Public Member Functions

- **throw\_value\_limit** (const [throw\\_value\\_limit](#) &\_\_other)
- **throw\_value\_limit** ([throw\\_value\\_limit](#) &&)=default
- **throw\_value\_limit** (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_limit](#) & **operator=** (const [throw\\_value\\_limit](#) &\_\_other)
- [throw\\_value\\_limit](#) & **operator=** ([throw\\_value\\_limit](#) &&)=default

## Static Public Member Functions

- static size\_t & **count** ()
- static size\_t & **limit** ()
- static void **set\_limit** (const size\_t \_\_l)
- static void **throw\_conditionally** ()

#### Public Attributes

- `std::size_t _M_i`

#### 4.964.1 Detailed Description

Type throwing via limit condition.

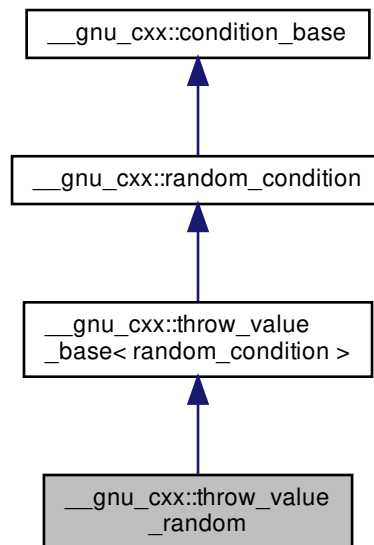
Definition at line 740 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

#### 4.965 `__gnu_cxx::throw_value_random` Struct Reference

Inheritance diagram for `__gnu_cxx::throw_value_random`:



#### Public Types

- typedef `throw_value_base< random_condition >` **base\_type**
- typedef `random_condition` **condition\_type**

## Public Member Functions

- **throw\_value\_random** (const [throw\\_value\\_random](#) &\_\_other)
- **throw\_value\_random** ([throw\\_value\\_random](#) &&)=default
- **throw\_value\_random** (const std::size\_t \_\_i)
- [throw\\_value\\_base](#) & **operator++** ()
- [throw\\_value\\_random](#) & **operator=** (const [throw\\_value\\_random](#) &\_\_other)
- [throw\\_value\\_random](#) & **operator=** ([throw\\_value\\_random](#) &&)=default
- void **seed** (unsigned long \_\_s)

## Static Public Member Functions

- static void **set\_probability** (double \_\_p)
- static void **throw\_conditionally** ()

## Public Attributes

- std::size\_t **M\_i**

## 4.965.1 Detailed Description

Type throwing via random condition.

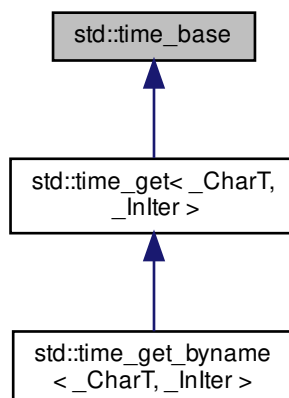
Definition at line 772 of file `throw_allocator.h`.

The documentation for this struct was generated from the following file:

- [throw\\_allocator.h](#)

## 4.966 std::time\_base Class Reference

Inheritance diagram for std::time\_base:



#### Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }

#### 4.966.1 Detailed Description

Time format ordering data.

This class provides an enum representing different orderings of time: day, month, and year.

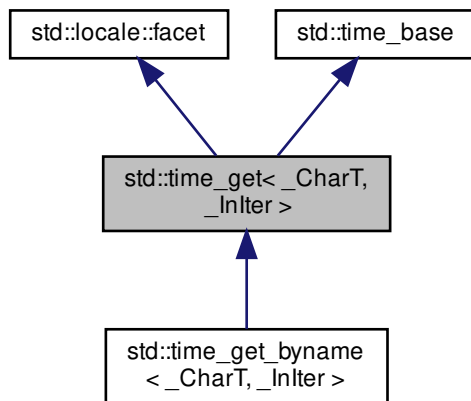
Definition at line 52 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

#### 4.967 std::time\_get<\_CharT, \_InIter> Class Template Reference

Inheritance diagram for `std::time_get<_CharT, _InIter>`:



#### Public Types

- enum **dateorder** {  
    **no\_order**, **dmy**, **mdy**, **ymd**,  
    **ydm** }

- typedef `_CharT` [char\\_type](#)
- typedef `_InIter` [iter\\_type](#)

## Public Member Functions

- [time\\_get](#) (size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- [iter\\_type get](#) (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier=0) const
- [iter\\_type get](#) (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const char\_type \* \_\_fmt, const char\_type \* \_\_fmtend) const
- [iter\\_type get\\_date](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- [iter\\_type get\\_monthname](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- [iter\\_type get\\_time](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- [iter\\_type get\\_weekday](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- [iter\\_type get\\_year](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [~time\\_get](#) ()
- [iter\\_type M\\_extract\\_name](#) (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- [iter\\_type M\\_extract\\_num](#) (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, int \_\_min, int \_\_max, size\_t \_\_len, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- [iter\\_type M\\_extract\\_via\\_format](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm, const \_CharT \* \_\_format) const
- [iter\\_type M\\_extract\\_wday\\_or\\_month](#) (iter\_type \_\_beg, iter\_type \_\_end, int & \_\_member, const \_CharT \*\* \_\_names, size\_t \_\_indexlen, ios\_base & \_\_io, ios\_base::iostate & \_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- [iter\\_type do\\_get](#) (iter\_type \_\_s, iter\_type \_\_end, ios\_base & \_\_f, ios\_base::iostate & \_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier) const
- virtual [iter\\_type do\\_get\\_date](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual [iter\\_type do\\_get\\_monthname](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual [iter\\_type do\\_get\\_time](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual [iter\\_type do\\_get\\_weekday](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const
- virtual [iter\\_type do\\_get\\_year](#) (iter\_type \_\_beg, iter\_type \_\_end, ios\_base & \_\_io, ios\_base::iostate & \_\_err, tm \* \_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc, const char \* \_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale & \_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \* \_\_s)

#### 4.967.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get< _CharT, _InIter >
```

Primary class template `time_get`.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.

The `time_get` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_get` facet.

Definition at line 368 of file `locale_facets_nonio.h`.

#### 4.967.2 Member Typedef Documentation

##### 4.967.2.1 `char_type`

```
template<typename _CharT , typename _InIter >
typedef _CharT std::time_get< _CharT, _InIter >::char_type
```

Public typedefs.

Definition at line 374 of file `locale_facets_nonio.h`.

##### 4.967.2.2 `iter_type`

```
template<typename _CharT , typename _InIter >
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
```

Public typedefs.

Definition at line 375 of file `locale_facets_nonio.h`.

#### 4.967.3 Constructor & Destructor Documentation

##### 4.967.3.1 `time_get()`

```
template<typename _CharT , typename _InIter >
std::time_get< _CharT, _InIter >::time_get (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.



#### Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 389 of file locale\_facets\_nonio.h.

#### 4.967.3.2 ~time\_get()

```
template<typename _CharT , typename _InIter >
virtual std::time_get< _CharT, _InIter >::~~time_get () [inline], [protected], [virtual]
```

Destructor.

Definition at line 593 of file locale\_facets\_nonio.h.

### 4.967.4 Member Function Documentation

#### 4.967.4.1 date\_order()

```
template<typename _CharT , typename _InIter >
dateorder std::time_get< _CharT, _InIter >::date_order () const [inline]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_get::put::put()` only uses month, day, and year. If the format `x` for the associated locale uses other fields, this function returns `time_base::dateorder::noorder`.

NOTE: The library always returns `noorder` at the moment.

#### Returns

A member of `time_base::dateorder`.

Definition at line 406 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_date_order()`.

#### 4.967.4.2 do\_date\_order()

```
template<typename _CharT , typename _InIter >
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↵
date_order () const [protected], [virtual]
```

Return preferred order of month, day, and year.

This function returns an enum from `time_base::dateorder` giving the preferred ordering if the format `x` given to `time_↵`  
`put::put()` only uses month, day, and year. This function is a hook for derived classes to change the value returned.

##### Returns

A member of `time_base::dateorder`.

Definition at line 627 of file `locale_facets_nonio.tcc`.

Referenced by `std::time_get< _CharT, _InIter >::date_order()`.

#### 4.967.4.3 do\_get()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 ios_base & __f,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier) const [inline], [protected]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

##### See also

`get()` for more details.

##### Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__f</code>        | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

**Returns**

Iterator to first char not parsed.

Definition at line 1241 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::goodbit, and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::time\_get< \_CharT, \_InIter >::get().

**4.967.4.4 do\_get\_date()**

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

get\_date() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond date string.

Definition at line 1077 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_date().

#### 4.967.4.5 do\_get\_monthname()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

#### See also

[get\\_monthname\(\)](#) for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond month name.

Definition at line 1120 of file locale\_facets\_nonio.tcc.

References [std::ios\\_base::\\_M\\_getloc\(\)](#), [std::ios\\_base::eofbit](#), [std::ios\\_base::failbit](#), and [std::ios\\_base::goodbit](#).

Referenced by [std::time\\_get< \\_CharT, \\_InIter >::get\\_monthname\(\)](#).

#### 4.967.4.6 do\_get\_time()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_time() for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond time string.

Definition at line 1060 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_time()`.

#### 4.967.4.7 do\_get\_weekday()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_weekday() for details.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1094 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_weekday()`.

**4.967.4.8 do\_get\_year()**

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

`get_year()` for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

Definition at line 1146 of file locale\_facets\_nonio.tcc.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_year()`.

## 4.967.4.9 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier = 0) const [inline]
```

Parse input string according to format.

This function calls time\_get::do\_get with the provided parameters.

## See also

do\_get() and get().

## Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

## Returns

Iterator to first char not parsed.

Definition at line 559 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get().

## 4.967.4.10 get() [2/2]

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
```

```
const char_type * __fmt,
const char_type * __fmtend) const [inline]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of `time_put::put`. The format string follows the format specified for `strptime(3)/strptime(3)`. The actual parsing is done by `time_get::do_get`.

#### Parameters

|                       |                                               |
|-----------------------|-----------------------------------------------|
| <code>__s</code>      | Start of string to parse.                     |
| <code>__end</code>    | End of string to parse.                       |
| <code>__io</code>     | Source of the locale.                         |
| <code>__err</code>    | Error flags to set.                           |
| <code>__tm</code>     | Pointer to struct <code>tm</code> to fill in. |
| <code>__fmt</code>    | Start of the format string.                   |
| <code>__fmtend</code> | End of the format string.                     |

#### Returns

Iterator to first char not parsed.

Definition at line 1169 of file `locale_facets_nonio.tcc`.

References `std::ios_base::M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::__ctype_abstract_base< _CharT >::is()`, `std::__ctype_abstract_base< _CharT >::narrow()`, `std::__ctype_abstract_base< _CharT >::tolower()`, and `std::__ctype_abstract_base< _CharT >::toupper()`.

#### 4.967.4.11 `get_date()`

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input date string.

This function parses a date according to the format `x` and puts the results into a user-supplied struct `tm`. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format `x`, `tm` will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err` |= `ios_base::failbit`. If parsing reads all the characters, `err` |= `ios_base::eofbit`.



## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond date string.

Definition at line 455 of file locale\_facets\_nonio.h.

References `std::time_get< _CharT, _InIter >::do_get_date()`.

## 4.967.4.12 get\_monthname()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_monthname()`.

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond month name.

Definition at line 512 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter >::do_get_monthname()`.

**4.967.4.13 get\_time()**

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 430 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter >::do_get_time()`.

## 4.967.4.14 get\_weekday()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond weekday name.

Definition at line 483 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_weekday().

## 4.967.4.15 get\_year()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling `time_get::do_get_year()`.

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

#### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

#### Returns

Iterator to first char beyond year.

Definition at line 538 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_year()`.

### 4.967.5 Member Data Documentation

#### 4.967.5.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::time_get< _CharT, _InIter >::id [static]
```

Numpunct facet id.

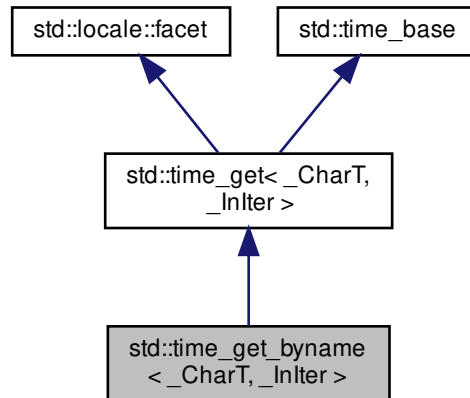
Definition at line 379 of file `locale_facets_nonio.h`.

The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)

## 4.968 std::time\_get\_byname&lt;\_CharT, \_InIter&gt; Class Template Reference

Inheritance diagram for std::time\_get\_byname<\_CharT, \_InIter>:



## Public Types

- typedef `_CharT` **char\_type**
- enum **dateorder** {  
  **no\_order**, **dmy**, **mdy**, **ymd**,  
  **ydm** }
- typedef `_InIter` **iter\_type**

## Public Member Functions

- **time\_get\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_get\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- dateorder [date\\_order](#) () const
- **iter\_type** get (iter\_type \_\_s, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm, char \_\_format, char \_\_modifier=0) const
- **iter\_type** get (iter\_type \_\_s, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm, const [char\\_type](#) \* \_\_fmt, const [char\\_type](#) \* \_\_fmtend) const
- **iter\_type** get\_date (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm) const
- **iter\_type** get\_monthname (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm) const
- **iter\_type** get\_time (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm) const
- **iter\_type** get\_weekday (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm) const
- **iter\_type** get\_year (iter\_type \_\_beg, iter\_type \_\_end, ios\_base &\_\_io, ios\_base::iostate &\_\_err, tm \* \_\_tm) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- [iter\\_type\\_M\\_extract\\_name](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type\\_M\\_extract\\_num](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, int \_\_min, int \_\_max, size\_t \_\_len, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- [iter\\_type\\_M\\_extract\\_via\\_format](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, const \_CharT \*\_\_format) const
- [iter\\_type\\_M\\_extract\\_wday\\_or\\_month](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, int &\_\_member, const \_CharT \*\*\_\_names, size\_t \_\_indexlen, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err) const
- virtual dateorder [do\\_date\\_order](#) () const
- [iter\\_type do\\_get](#) ([iter\\_type](#) \_\_s, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_f, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm, char \_\_format, char \_\_modifier) const
- virtual [iter\\_type do\\_get\\_date](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_monthname](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_time](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_weekday](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const
- virtual [iter\\_type do\\_get\\_year](#) ([iter\\_type](#) \_\_beg, [iter\\_type](#) \_\_end, [ios\\_base](#) &\_\_io, [ios\\_base::iostate](#) &\_\_err, tm \*\_\_tm) const

## Static Protected Member Functions

- static \_\_c\_locale [\\_S\\_clone\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc) throw ()
- static void [\\_S\\_create\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc, const char \*\_\_s, \_\_c\_locale \_\_old=0)
- static void [\\_S\\_destroy\\_c\\_locale](#) (\_\_c\_locale &\_\_cloc)
- static \_\_c\_locale [\\_S\\_get\\_c\\_locale](#) ()
- static const char \* [\\_S\\_get\\_c\\_name](#) () throw ()
- static \_\_c\_locale [\\_S\\_lc\\_ctype\\_c\\_locale](#) (\_\_c\_locale \_\_cloc, const char \*\_\_s)

## 4.968.1 Detailed Description

```
template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >
```

class time\_get\_byname [22.2.5.2].

Definition at line 760 of file locale\_facets\_nonio.h.

## 4.968.2 Member Function Documentation

### 4.968.2.1 date\_order()

```
template<typename _CharT , typename _InIter >
dateorder std::time_get< _CharT, _InIter >::date_order () const [inline], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_get::put() only uses month, day, and year. If the format x for the associated locale uses other fields, this function returns time\_base::dateorder::noorder.

NOTE: The library always returns noorder at the moment.

#### Returns

A member of time\_base::dateorder.

Definition at line 406 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_date\_order().

### 4.968.2.2 do\_date\_order()

```
template<typename _CharT , typename _InIter >
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_
date_order () const [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.

This function returns an enum from time\_base::dateorder giving the preferred ordering if the format x given to time\_get::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

#### Returns

A member of time\_base::dateorder.

Definition at line 627 of file locale\_facets\_nonio.tcc.

Referenced by std::time\_get< \_CharT, \_InIter >::date\_order().

#### 4.968.2.3 do\_get()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get (
 iter_type __s,
 iter_type __end,
 ios_base & __f,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier) const [inline], [protected], [inherited]
```

Parse input string according to format.

This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

`get()` for more details.

#### Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__f</code>        | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

#### Returns

Iterator to first char not parsed.

Definition at line 1241 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::goodbit`, and `std::__ctype_abstract_base<_CharT >::widen()`.

Referenced by `std::time_get<_CharT, _InIter >::get()`.



## 4.968.2.4 do\_get\_date()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input date string.

This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get\_date() for details.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond date string.

Definition at line 1077 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, and `std::ios_base::eofbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_date()`.

## 4.968.2.5 do\_get\_monthname()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_monthname()` for details.

Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

Returns

Iterator to first char beyond month name.

Definition at line 1120 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get<_CharT, _InIter>::get_monthname()`.

#### 4.968.2.6 `do_get_time()`

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input time string.

This function parses a time according to the format `x` and puts the results into a user-supplied struct `tm`. This function is a hook for derived classes to change the value returned.

See also

`get_time()` for details.

Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond time string.

Definition at line 1060 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::ios\_base::eofbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_time().

**4.968.2.7 do\_get\_weekday()**

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

**See also**

get\_weekday() for details.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond weekday name.

Definition at line 1094 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), std::ios\_base::eofbit, std::ios\_base::failbit, and std::ios\_base::goodbit.

Referenced by std::time\_get< \_CharT, \_InIter >::get\_weekday().

#### 4.968.2.8 do\_get\_year()

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::do_get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [protected], [virtual], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

`get_year()` for details.

##### Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

##### Returns

Iterator to first char beyond year.

Definition at line 1146 of file `locale_facets_nonio.tcc`.

References `std::ios_base::eofbit`, `std::ios_base::failbit`, and `std::ios_base::goodbit`.

Referenced by `std::time_get< _CharT, _InIter >::get_year()`.

#### 4.968.2.9 get() [1/2]

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 char __format,
 char __modifier = 0) const [inline], [inherited]
```

Parse input string according to format.

This function calls `time_get::do_get` with the provided parameters.

See also

do\_get() and get().

#### Parameters

|                         |                                  |
|-------------------------|----------------------------------|
| <code>__s</code>        | Start of string to parse.        |
| <code>__end</code>      | End of string to parse.          |
| <code>__io</code>       | Source of the locale.            |
| <code>__err</code>      | Error flags to set.              |
| <code>__tm</code>       | Pointer to struct tm to fill in. |
| <code>__format</code>   | Format specifier.                |
| <code>__modifier</code> | Format modifier.                 |

#### Returns

Iterator to first char not parsed.

Definition at line 559 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get().

#### 4.968.2.10 get() [2/2]

```
template<typename _CharT , typename _InIter >
_InIter std::time_get< _CharT, _InIter >::get (
 iter_type __s,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm,
 const char_type * __fmt,
 const char_type * __fmtend) const [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of time\_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time\_get::do\_get.

#### Parameters

|                       |                                  |
|-----------------------|----------------------------------|
| <code>__s</code>      | Start of string to parse.        |
| <code>__end</code>    | End of string to parse.          |
| <code>__io</code>     | Source of the locale.            |
| <code>__err</code>    | Error flags to set.              |
| <code>__tm</code>     | Pointer to struct tm to fill in. |
| <code>__fmt</code>    | Start of the format string.      |
| <code>__fmtend</code> | End of the format string.        |

**Returns**

Iterator to first char not parsed.

Definition at line 1169 of file locale\_facets\_nonio.tcc.

References `std::ios_base::_M_getloc()`, `std::ios_base::eofbit`, `std::ios_base::failbit`, `std::ios_base::goodbit`, `std::__ctype_abstract_base<_CharT>::is()`, `std::__ctype_abstract_base<_CharT>::narrow()`, `std::__ctype_abstract_base<_CharT>::tolower()`, and `std::__ctype_abstract_base<_CharT>::toupper()`.

**4.968.2.11 get\_date()**

```
template<typename _CharT, typename _InIter >
iter_type std::time_get<_CharT, _InIter>::get_date (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_date()`.

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, `err |= ios_base::failbit`. If parsing reads all the characters, `err |= ios_base::eofbit`.

**Parameters**

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

**Returns**

Iterator to first char beyond date string.

Definition at line 455 of file locale\_facets\_nonio.h.

References `std::time_get<_CharT, _InIter>::do_get_date()`.

## 4.968.2.12 get\_monthname()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_monthname (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input month string.

This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_monthname().

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

## Parameters

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

## Returns

Iterator to first char beyond month name.

Definition at line 512 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_monthname().

## 4.968.2.13 get\_time()

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_time (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_time()`.

If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |

#### Returns

Iterator to first char beyond time string.

Definition at line 430 of file `locale_facets_nonio.h`.

References `std::time_get<_CharT, _InIter>::do_get_time()`.

#### 4.968.2.14 `get_weekday()`

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_weekday (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct *tm*. The result is returned by calling `time_get::do_get_weekday()`.

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, `err != ios_base::failbit`. If parsing reads all the characters, `err != ios_base::eofbit`.

#### Parameters

|                    |                                         |
|--------------------|-----------------------------------------|
| <code>__beg</code> | Start of string to parse.               |
| <code>__end</code> | End of string to parse.                 |
| <code>__io</code>  | Source of the locale.                   |
| <code>__err</code> | Error flags to set.                     |
| <code>__tm</code>  | Pointer to struct <i>tm</i> to fill in. |



**Returns**

Iterator to first char beyond weekday name.

Definition at line 483 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_weekday().

**4.968.2.15 get\_year()**

```
template<typename _CharT , typename _InIter >
iter_type std::time_get< _CharT, _InIter >::get_year (
 iter_type __beg,
 iter_type __end,
 ios_base & __io,
 ios_base::iostate & __err,
 tm * __tm) const [inline], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time\_get::do\_get\_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios\_base::failbit. If parsing reads all the characters, err |= ios\_base::eofbit.

**Parameters**

|                    |                                  |
|--------------------|----------------------------------|
| <code>__beg</code> | Start of string to parse.        |
| <code>__end</code> | End of string to parse.          |
| <code>__io</code>  | Source of the locale.            |
| <code>__err</code> | Error flags to set.              |
| <code>__tm</code>  | Pointer to struct tm to fill in. |

**Returns**

Iterator to first char beyond year.

Definition at line 538 of file locale\_facets\_nonio.h.

References std::time\_get< \_CharT, \_InIter >::do\_get\_year().

**4.968.3 Member Data Documentation**

## 4.968.3.1 id

```
template<typename _CharT , typename _InIter >
locale::id std::time_get< _CharT, _InIter >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 379 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.969 std::chrono::time\_point&lt; \_Clock, \_Dur &gt; Struct Template Reference

## Public Types

- typedef \_Clock **clock**
- typedef \_Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

## Public Member Functions

- constexpr **time\_point** (const duration &\_\_dur)
- template<typename \_Dur2, typename = \_Require<is\_convertible<\_Dur2, \_Dur>>>>  
constexpr **time\_point** (const [time\\_point](#)< clock, \_Dur2 > &\_\_t)
- constexpr [time\\_point](#) & **operator+=** (const duration &\_\_dur)
- constexpr [time\\_point](#) & **operator-=** (const duration &\_\_dur)
- constexpr duration **time\_since\_epoch** () const

## Static Public Member Functions

- static constexpr [time\\_point](#) **max** () noexcept
- static constexpr [time\\_point](#) **min** () noexcept

## Related Functions

(Note that these are not member functions.)

- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2 >  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< \_Dur1, duration< \_Rep2, \_Period2 > >::type >  
[operator+](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Rep1, typename \_Period1, typename \_Clock, typename \_Dur2 >  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< duration< \_Rep1, \_Period1 >, \_Dur2 >::type >  
[operator+](#) (const duration< \_Rep1, \_Period1 > &\_\_lhs, const [time\\_point](#)< \_Clock, \_Dur2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Rep2, typename \_Period2 >  
constexpr [time\\_point](#)< \_Clock, typename [common\\_type](#)< \_Dur1, duration< \_Rep2, \_Period2 > >::type >  
[operator-](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const duration< \_Rep2, \_Period2 > &\_\_rhs)
- template<typename \_Clock, typename \_Dur1, typename \_Dur2 >  
constexpr [common\\_type](#)< \_Dur1, \_Dur2 >::type [operator-](#) (const [time\\_point](#)< \_Clock, \_Dur1 > &\_\_lhs, const  
[time\\_point](#)< \_Clock, \_Dur2 > &\_\_rhs)

## 4.969.1 Detailed Description

```
template<typename _Clock, typename _Dur>
struct std::chrono::time_point< _Clock, _Dur >
```

time\_point

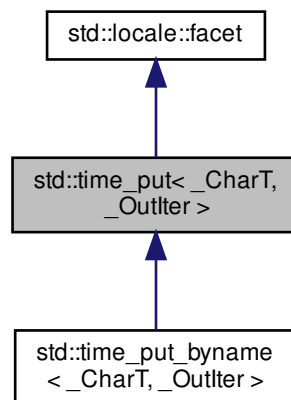
Definition at line 74 of file chrono.

The documentation for this struct was generated from the following file:

- [chrono](#)

## 4.970 std::time\_put&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for std::time\_put< \_CharT, \_Outlter >:



## Public Types

- typedef \_CharT [char\\_type](#)
- typedef \_Outlter [iter\\_type](#)

### Public Member Functions

- `time_put` (size\_t \_\_refs=0)
- `iter_type put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, const \_CharT \*\_\_beg, const \_CharT \*\_\_end) const
- `iter_type put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

### Static Public Attributes

- static `locale::id` id

### Protected Member Functions

- virtual `~time_put` ()
- virtual `iter_type do_put` (iter\_type \_\_s, ios\_base &\_\_io, char\_type \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

### Static Protected Member Functions

- static `_c_locale _S_clone_c_locale` (\_c\_locale &\_\_cloc) throw ()
- static void `_S_create_c_locale` (\_c\_locale &\_\_cloc, const char \*\_\_s, \_c\_locale \_\_old=0)
- static void `_S_destroy_c_locale` (\_c\_locale &\_\_cloc)
- static `_c_locale _S_get_c_locale` ()
- static const char \* `_S_get_c_name` () throw ()
- static `_c_locale _S_lc_ctype_c_locale` (\_c\_locale \_\_cloc, const char \*\_\_s)

#### 4.970.1 Detailed Description

```
template<typename _CharT, typename _Outiter>
class std::time_put< _CharT, _Outiter >
```

Primary class template `time_put`.

This facet encapsulates the code to format and output dates and times according to formats used by `strftime()`.

The `time_put` template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the `time_put` facet.

Definition at line 797 of file `locale_facets_nonio.h`.

#### 4.970.2 Member Typedef Documentation

#### 4.970.2.1 char\_type

```
template<typename _CharT , typename _OutIter >
typedef _CharT std::time_put< _CharT, _OutIter >::char_type
```

Public typedefs.

Definition at line 803 of file locale\_facets\_nonio.h.

#### 4.970.2.2 iter\_type

```
template<typename _CharT , typename _OutIter >
typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

Definition at line 804 of file locale\_facets\_nonio.h.

### 4.970.3 Constructor & Destructor Documentation

#### 4.970.3.1 time\_put()

```
template<typename _CharT , typename _OutIter >
std::time_put< _CharT, _OutIter >::time_put (
 size_t __refs = 0) [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

|                     |                                 |
|---------------------|---------------------------------|
| <code>__refs</code> | Passed to the base facet class. |
|---------------------|---------------------------------|

Definition at line 818 of file locale\_facets\_nonio.h.

#### 4.970.3.2 ~time\_put()

```
template<typename _CharT , typename _OutIter >
virtual std::time_put< _CharT, _OutIter >::~~time_put () [inline], [protected], [virtual]
```

Destructor.

Definition at line 864 of file locale\_facets\_nonio.h.

#### 4.970.4 Member Function Documentation

##### 4.970.4.1 do\_put()

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod) const [protected], [virtual]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

##### See also

`put()` for more details.

##### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | <code>char_type</code> to use for padding.   |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

##### Returns

Iterator after writing.

Definition at line 1309 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::widen()`.

Referenced by `std::time_put< _CharT, _OutIter >::put()`.

## 4.970.4.2 put() [1/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 const _CharT * __beg,
 const _CharT * __end) const
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by `strftime()`.

## Parameters

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The stream to write to.                      |
| <code>__io</code>   | Source of locale.                            |
| <code>__fill</code> | <code>char_type</code> to use for padding.   |
| <code>__tm</code>   | Struct tm with date and time info to format. |
| <code>__beg</code>  | Start of format string.                      |
| <code>__end</code>  | End of format string.                        |

## Returns

Iterator after writing.

Definition at line 1274 of file `locale_facets_nonio.tcc`.

References `std::ios_base::_M_getloc()`, and `std::__ctype_abstract_base< _CharT >::narrow()`.

## 4.970.4.3 put() [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod = 0) const [inline]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by `strftime()`. It does so by returning `time_put::do_put()`.

**Parameters**

|                       |                                                           |
|-----------------------|-----------------------------------------------------------|
| <code>__s</code>      | The stream to write to.                                   |
| <code>__io</code>     | Source of locale.                                         |
| <code>__fill</code>   | <code>char_type</code> to use for padding.                |
| <code>__tm</code>     | Struct <code>tm</code> with date and time info to format. |
| <code>__format</code> | Format char.                                              |
| <code>__mod</code>    | Optional modifier char.                                   |

**Returns**

Iterator after writing.

Definition at line 857 of file `locale_facets_nonio.h`.

References `std::time_put<_CharT, _OutIter >::do_put()`.

**4.970.5 Member Data Documentation****4.970.5.1 id**

```
template<typename _CharT , typename _OutIter >
locale::id std::time_put<_CharT, _OutIter >::id [static]
```

Numpunct facet id.

Definition at line 808 of file `locale_facets_nonio.h`.

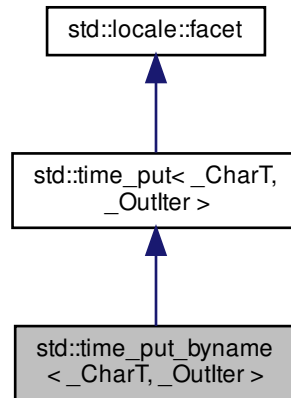
The documentation for this class was generated from the following files:

- [locale\\_facets\\_nonio.h](#)
- [locale\\_facets\\_nonio.tcc](#)



## 4.971 std::time\_put\_byname&lt; \_CharT, \_Outlter &gt; Class Template Reference

Inheritance diagram for std::time\_put\_byname< \_CharT, \_Outlter >:



## Public Types

- typedef `_CharT` **char\_type**
- typedef `_Outlter` **iter\_type**

## Public Member Functions

- **time\_put\_byname** (const char \*, size\_t \_\_refs=0)
- **time\_put\_byname** (const [string](#) &\_\_s, size\_t \_\_refs=0)
- [iter\\_type](#) **put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, const `_CharT` \*\_\_beg, const `_CharT` \*\_\_end) const
- [iter\\_type](#) **put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod=0) const

## Static Public Attributes

- static [locale::id](#) id

## Protected Member Functions

- virtual [iter\\_type](#) **do\_put** ([iter\\_type](#) \_\_s, [ios\\_base](#) &\_\_io, [char\\_type](#) \_\_fill, const tm \*\_\_tm, char \_\_format, char \_\_mod) const

## Static Protected Member Functions

- static `__c_locale _S_clone_c_locale (__c_locale &__cloc) throw ()`
- static void `_S_create_c_locale (__c_locale &__cloc, const char *__s, __c_locale __old=0)`
- static void `_S_destroy_c_locale (__c_locale &__cloc)`
- static `__c_locale _S_get_c_locale ()`
- static const char \* `_S_get_c_name () throw ()`
- static `__c_locale _S_lc_ctype_c_locale (__c_locale __cloc, const char *__s)`

### 4.971.1 Detailed Description

```
template<typename _CharT, typename _OutIter>
class std::time_put_byname< _CharT, _OutIter >
```

class time\_put\_byname [22.2.5.4].

Definition at line 893 of file locale\_facets\_nonio.h.

### 4.971.2 Member Function Documentation

#### 4.971.2.1 do\_put()

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::do_put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod) const [protected], [virtual], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

#### See also

`put()` for more details.

#### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

**Returns**

Iterator after writing.

Definition at line 1309 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::widen().

Referenced by std::time\_put< \_CharT, \_OutIter >::put().

**4.971.2.2 put()** [1/2]

```
template<typename _CharT , typename _OutIter >
_OutIter std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 const _CharT * __beg,
 const _CharT * __end) const [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

**Parameters**

|                     |                                              |
|---------------------|----------------------------------------------|
| <code>__s</code>    | The stream to write to.                      |
| <code>__io</code>   | Source of locale.                            |
| <code>__fill</code> | char_type to use for padding.                |
| <code>__tm</code>   | Struct tm with date and time info to format. |
| <code>__beg</code>  | Start of format string.                      |
| <code>__end</code>  | End of format string.                        |

**Returns**

Iterator after writing.

Definition at line 1274 of file locale\_facets\_nonio.tcc.

References std::ios\_base::\_M\_getloc(), and std::\_\_ctype\_abstract\_base< \_CharT >::narrow().

#### 4.971.2.3 put() [2/2]

```
template<typename _CharT , typename _OutIter >
iter_type std::time_put< _CharT, _OutIter >::put (
 iter_type __s,
 ios_base & __io,
 char_type __fill,
 const tm * __tm,
 char __format,
 char __mod = 0) const [inline], [inherited]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time\_put::do\_put().

##### Parameters

|                       |                                              |
|-----------------------|----------------------------------------------|
| <code>__s</code>      | The stream to write to.                      |
| <code>__io</code>     | Source of locale.                            |
| <code>__fill</code>   | char_type to use for padding.                |
| <code>__tm</code>     | Struct tm with date and time info to format. |
| <code>__format</code> | Format char.                                 |
| <code>__mod</code>    | Optional modifier char.                      |

##### Returns

Iterator after writing.

Definition at line 857 of file locale\_facets\_nonio.h.

References std::time\_put< \_CharT, \_OutIter >::do\_put().

#### 4.971.3 Member Data Documentation

##### 4.971.3.1 id

```
template<typename _CharT , typename _OutIter >
locale::id std::time_put< _CharT, _OutIter >::id [static], [inherited]
```

Numpunct facet id.

Definition at line 808 of file locale\_facets\_nonio.h.

The documentation for this class was generated from the following file:

- [locale\\_facets\\_nonio.h](#)

## 4.972 `std::timed_mutex` Class Reference

### Public Member Functions

- **timed\_mutex** (const [timed\\_mutex](#) &)=delete
- void **lock** ()
- [timed\\_mutex](#) & **operator=** (const [timed\\_mutex](#) &)=delete
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

### 4.972.1 Detailed Description

`timed_mutex`

Definition at line 343 of file `mutex`.

The documentation for this class was generated from the following file:

- [mutex](#)

## 4.973 `std::to_chars_result` Struct Reference

### Public Attributes

- `errc` **ec**
- `char *` **ptr**

### 4.973.1 Detailed Description

Result type of `std::to_chars`.

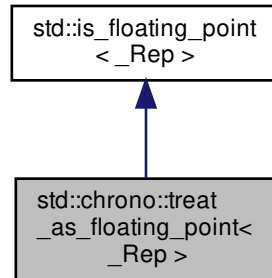
Definition at line 55 of file `charconv`.

The documentation for this struct was generated from the following file:

- [charconv](#)

#### 4.974 `std::chrono::treat_as_floating_point<_Rep>` Struct Template Reference

Inheritance diagram for `std::chrono::treat_as_floating_point<_Rep>`:



##### 4.974.1 Detailed Description

```
template<typename _Rep>
struct std::chrono::treat_as_floating_point<_Rep>
```

`treat_as_floating_point`

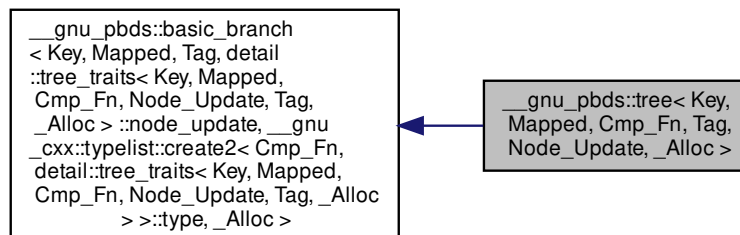
Definition at line 268 of file `chrono`.

The documentation for this struct was generated from the following file:

- [chrono](#)

#### 4.975 `__gnu_pbds::tree<Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc>` Class Template Reference

Inheritance diagram for `__gnu_pbds::tree<Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc>`:



## Public Types

- typedef Cmp\_Fn [cmp\\_fn](#)
- typedef [detail::tree\\_traits](#)< Key, Mapped, Cmp\_Fn, Node\_Update, Tag, \_Alloc > ::node\_update **node\_update**

## Public Member Functions

- [tree](#) (const [cmp\\_fn](#) &c)
- template<typename It >  
[tree](#) (It first, It last)
- template<typename It >  
[tree](#) (It first, It last, const [cmp\\_fn](#) &c)
- **tree** (const [tree](#) &other)
- [tree](#) & **operator=** (const [tree](#) &other)
- void **swap** ([tree](#) &other)

## 4.975.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename
Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update = null_node_update, typename _Alloc =
std::allocator<char>>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
```

A tree-based container.

## Template Parameters

|                    |                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                                |
| <i>Mapped</i>      | Map type.                                                                                                                                |
| <i>Cmp_Fn</i>      | Comparison functor.                                                                                                                      |
| <i>Tag</i>         | Instantiating data structure type, see <a href="#">container_tag</a> .                                                                   |
| <i>Node_Update</i> | Updates tree internal-nodes, restores invariants when invalidated. XXX See <a href="#">design::tree-based-containersnode</a> invariants. |
| <i>_Alloc</i>      | Allocator type.                                                                                                                          |

Base tag choices are: [ov\\_tree\\_tag](#), [rb\\_tree\\_tag](#), [splay\\_tree\\_tag](#).

Base is [basic\\_branch](#).

Definition at line 635 of file [assoc\\_container.hpp](#).

## 4.975.2 Member Typedef Documentation

#### 4.975.2.1 cmp\_fn

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
```

Comparison functor type.

Definition at line 642 of file assoc\_container.hpp.

#### 4.975.3 Constructor & Destructor Documentation

##### 4.975.3.1 tree() [1/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 const cmp_fn & c) [inline]
```

Constructor taking some policy objects. r\_cmp\_fn will be copied by the Cmp\_Fn object of the container object.

Definition at line 648 of file assoc\_container.hpp.

##### 4.975.3.2 tree() [2/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It >
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 It first,
 It last) [inline]
```

Constructor taking \_\_iterators to a range of value\_types. The value\_types between first\_it and last\_it will be inserted into the container object.

Definition at line 655 of file assoc\_container.hpp.



4.975.3.3 `tree()` [3/3]

```
template<typename Key , typename Mapped , typename Cmp_Fn = std::less<Key>, typename Tag = rb←
_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
 It first,
 It last,
 const cmp_fn & c) [inline]
```

Constructor taking \_\_iterators to a range of value\_types and some policy objects The value\_types between first\_it and last\_it will be inserted into the container object. r\_cmp\_fn will be copied by the cmp\_fn object of the container object.

Definition at line 663 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.976 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >` Struct Template Reference

## 4.976.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >
```

Tree metadata helper.

Definition at line 58 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

4.977 `__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >` Struct Template Reference

## Public Types

- `typedef Node_Update::metadata_type type`

#### 4.977.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.978 \_\_gnu\_pbds::detail::tree\_metadata\_helper< Node\_Update, true > Struct Template Reference

##### Public Types

- typedef [null\\_type](#) **type**

#### 4.978.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file tree\_policy/node\_metadata\_selector.hpp.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.979 \_\_gnu\_pbds::detail::tree\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc > Struct Template Reference

##### Public Types

- typedef [tree\\_metadata\\_helper](#)< \_\_node\_u, null\_update >::type **type**

#### 4.979.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename
Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Tree node metadata dispatch.

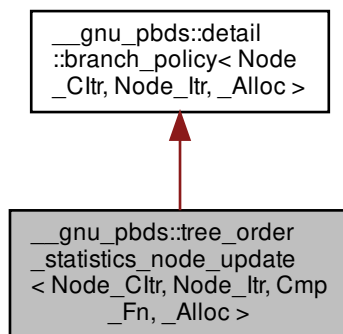
Definition at line 84 of file `tree_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [tree\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.980 `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc > Class` Template Reference

Inheritance diagram for `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`:



#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `Cmp_Fn` **cmp\_fn**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `size_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

#### Public Member Functions

- const\_iterator [find\\_by\\_order](#) (size\_type) const
- iterator [find\\_by\\_order](#) (size\_type)
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const

#### Protected Member Functions

- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

#### Private Types

- typedef Node\_Itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rkkey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef rebind\_traits< \_Alloc, rkkey\_type > **rebind\_k**
- typedef rebind\_traits< \_Alloc, rcvalue\_type > **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef [std::iterator\\_traits](#)< it\_type >::value\_type **value\_type**

#### Private Member Functions

- virtual it\_type **end** ()=0
- it\_type **end\_iterator** () const

#### Static Private Member Functions

- static key\_const\_reference **extract\_key** (const\_reference r\_val)

#### 4.980.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 64 of file tree\_policy.hpp.

#### 4.980.2 Member Function Documentation

#### 4.980.2.1 `find_by_order()` [1/2]

```
template<typename Node_CIttr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
__gnu_pbds::tree_order_statistics_node_update< Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_statistics_node_update<
Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
 size_type order) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 74 of file `tree_policy.hpp`.

#### 4.980.2.2 `find_by_order()` [2/2]

```
template<typename Node_CIttr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
__gnu_pbds::tree_order_statistics_node_update< Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statistics_node_update<
Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
 size_type order) [inline]
```

Finds an entry by `__order`. Returns an `iterator` to the entry with the `__order` order, or an `iterator` to the container object's end if order is at least the size of the container object.

Definition at line 47 of file `tree_policy.hpp`.

#### 4.980.2.3 `operator()()`

```
template<typename Node_CIttr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
void __gnu_pbds::tree_order_statistics_node_update< Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::operator()
(
 node_iterator node_it,
 node_const_iterator end_nd_it) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 110 of file `tree_policy.hpp`.

#### 4.980.2.4 `order_of_key()`

```
template<typename Node_CIttr , typename Node_Itr , typename Cmp_Fn , typename _Alloc >
__gnu_pbds::tree_order_statistics_node_update< Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statistics_node_update<
Node_CIttr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
 key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

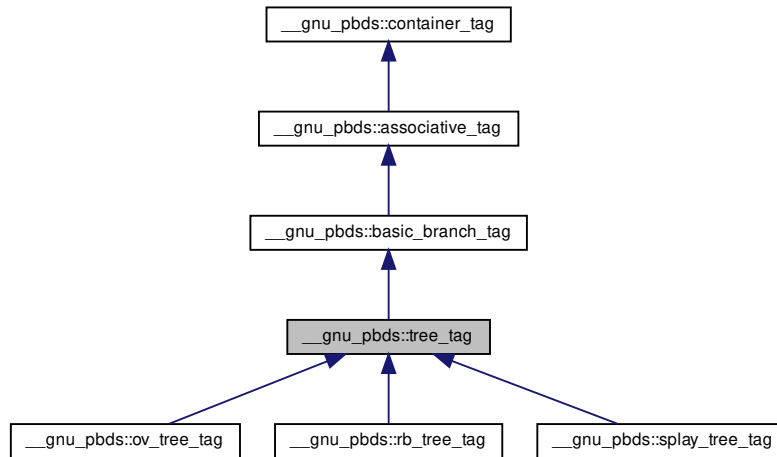
Definition at line 80 of file `tree_policy.hpp`.

The documentation for this class was generated from the following file:

- [tree\\_policy.hpp](#)

#### 4.981 `__gnu_pbds::tree_tag` Struct Reference

Inheritance diagram for `__gnu_pbds::tree_tag`:



##### 4.981.1 Detailed Description

Basic tree structure.

Definition at line 150 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

#### 4.982 `__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >` Struct Template Reference

##### 4.982.1 Detailed Description

```

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_
Fn, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >

```

Tree traits class, primary template.

Definition at line 70 of file `branch_policy/traits.hpp`.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

## 4.983 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` **node\_const\_iterator**
- typedef `ov_tree_node_it_< value_type, metadata_type, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_`  
`_update_pointer`

### 4.983.1 Detailed Description

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename
Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Tree traits.

Definition at line 61 of file `ov_tree_map_/traits.hpp`.

### 4.983.2 Member Typedef Documentation

#### 4.983.2.1 `node_const_iterator`

```
template<typename Key , typename Mapped , class Cmp_Fn , template< typename Node_Cltr, class
Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

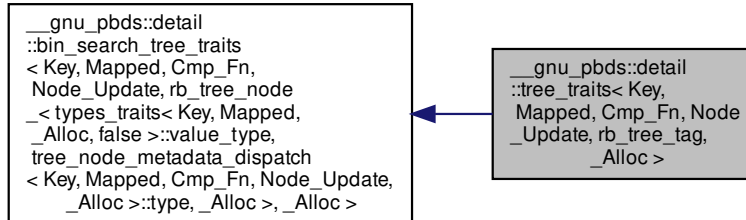
Definition at line 95 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

#### 4.984 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >`:



#### Public Types

- typedef `bin_search_tree_const_it< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **const\_reverse\_iterator**
- typedef `rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >` **node**
- typedef `bin_search_tree_const_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_const\_iterator**
- typedef `bin_search_tree_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`
- typedef `bin_search_tree_const_it< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_const\_iterator**
- typedef `bin_search_tree_it< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc >` **point\_iterator**
- typedef `bin_search_tree_it< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc >` **reverse\_iterator**

#### 4.984.1 Detailed Description

```

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

Definition at line 61 of file `rb_tree_map_traits.hpp`.



#### 4.984.2 Member Typedef Documentation

##### 4.984.2.1 `node_const_iterator`

```
typedef bin_search_tree_const_node_it< rb_tree_node< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↵
Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node< types_traits< Key, Mapped, _Alloc, false >↵
::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↵
Alloc > , _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

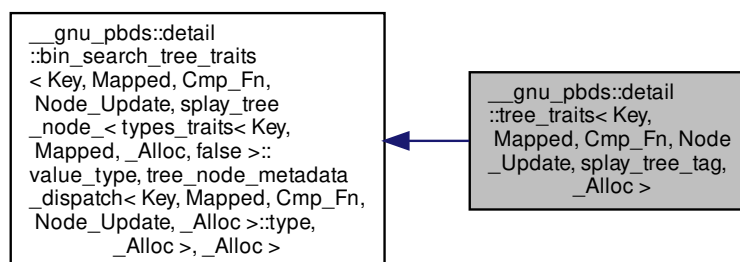
Definition at line 128 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

#### 4.985 `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc > **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > **node\_const\_iterator**
- typedef [bin\\_search\\_tree\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, Mapped, \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, Mapped, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **reverse\_iterator**

## 4.985.1 Detailed Description

```
template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 61 of file [splay\\_tree\\_/traits.hpp](#).

## 4.985.2 Member Typedef Documentation

## 4.985.2.1 node\_const\_iterator

```
typedef bin_search_tree_const_node_it < splay_tree_node < types_traits < Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_traits < Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node < types_traits < Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch < Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 128 of file [bin\\_search\\_tree\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

## 4.986 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `ov_tree_node_const_it_< value_type, metadata_type, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_const_iterator, Cmp_Fn, _Alloc >` **node\_update**
- typedef `__gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * null_node_update_pointer`

### 4.986.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
```

Specialization.

Definition at line 132 of file `ov_tree_map_/traits.hpp`.

### 4.986.2 Member Typedef Documentation

#### 4.986.2.1 `node_const_iterator`

```
template<typename Key , class Cmp_Fn , template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

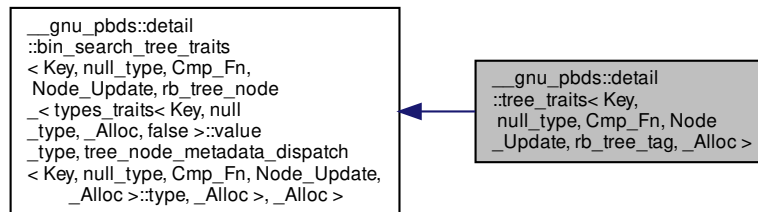
Definition at line 166 of file `ov_tree_map_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [ov\\_tree\\_map\\_/traits.hpp](#)

#### 4.987 \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc > Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >:



#### Public Types

- typedef `bin_search_tree_const_it` < typename node\_alloc\_traits::pointer, typename `type_traits::value_type`, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **const\_reverse\_iterator**
- typedef `rb_tree_node` < `types_traits` < Key, `null_type`, \_Alloc, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc > **node**
- typedef `bin_search_tree_const_node_it` < `rb_tree_node` < `types_traits` < Key, `null_type`, \_Alloc, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, `point_const_iterator`, `point_iterator`, \_Alloc > **node\_const\_iterator**
- typedef `bin_search_tree_node_it` < `rb_tree_node` < `types_traits` < Key, `null_type`, \_Alloc, false >::value\_type, `tree_node_metadata_dispatch` < Key, `null_type`, Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, `point_const_iterator`, `point_iterator`, \_Alloc > **node\_iterator**
- typedef Node\_Update < `node_const_iterator`, `node_iterator`, Cmp\_Fn, \_Alloc > **node\_update**
- typedef `__gnu_pbds::null_node_update` < `node_const_iterator`, `node_iterator`, Cmp\_Fn, \_Alloc > \* **null\_node\_update\_pointer**
- typedef `bin_search_tree_const_it` < typename node\_alloc\_traits::pointer, typename `type_traits::value_type`, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_const\_iterator**
- typedef `bin_search_tree_it` < typename node\_alloc\_traits::pointer, typename `type_traits::value_type`, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, true, \_Alloc > **point\_iterator**
- typedef `bin_search_tree_it` < typename node\_alloc\_traits::pointer, typename `type_traits::value_type`, typename type\_traits::pointer, typename type\_traits::const\_pointer, typename type\_traits::reference, typename type\_traits::const\_reference, false, \_Alloc > **reverse\_iterator**

#### 4.987.1 Detailed Description

```

template<typename Key, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

```

Specialization.

Definition at line 85 of file `rb_tree_map_/traits.hpp`.

## 4.987.2 Member Typedef Documentation

### 4.987.2.1 `node_const_iterator`

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↔
::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type , Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc > , _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

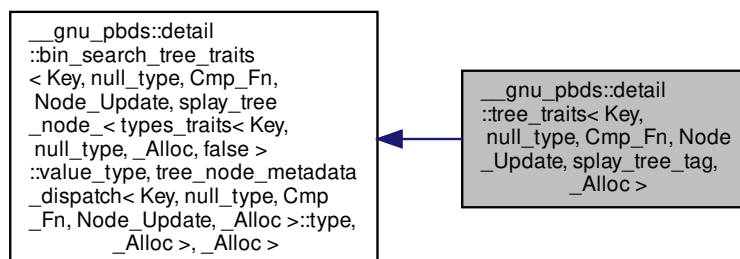
Definition at line 128 of file `bin_search_tree_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [rb\\_tree\\_map\\_/traits.hpp](#)

## 4.988 `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >` Struct Template Reference

Inheritance diagram for `__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >`:



## Public Types

- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename [type\\_traits::pointer](#), typename [type\\_traits::const\\_pointer](#), typename [type\\_traits::reference](#), typename [type\\_traits::const\\_reference](#), false, \_Alloc > **const\_reverse\_iterator**
- typedef [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), \_Alloc, false >::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc > **node**
- typedef [bin\\_search\\_tree\\_const\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), \_Alloc, false >↵::value\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > **node\_const\_iterator**
- typedef [bin\\_search\\_tree\\_node\\_it](#) < [splay\\_tree\\_node](#) < [types\\_traits](#) < Key, [null\\_type](#), \_Alloc, false >::value↵\_type, [tree\\_node\\_metadata\\_dispatch](#) < Key, [null\\_type](#), Cmp\_Fn, Node\_Update, \_Alloc >::type, \_Alloc >, [point\\_const\\_iterator](#), [point\\_iterator](#), \_Alloc > **node\_iterator**
- typedef Node\_Update < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > **node\_update**
- typedef [\\_\\_gnu\\_pbds::null\\_node\\_update](#) < [node\\_const\\_iterator](#), [node\\_iterator](#), Cmp\_Fn, \_Alloc > \* **null\_node↵\_update\_pointer**
- typedef [bin\\_search\\_tree\\_const\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename [type\\_traits::pointer](#), typename [type\\_traits::const\\_pointer](#), typename [type\\_traits::reference](#), typename [type\\_traits::const\\_reference](#), true, \_Alloc > **point\_const\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename [type\\_traits::pointer](#), typename [type\\_traits::const\\_pointer](#), typename [type\\_traits::reference](#), typename [type\\_traits↵::const\\_reference](#), true, \_Alloc > **point\_iterator**
- typedef [bin\\_search\\_tree\\_it](#) < typename node\_alloc\_traits::pointer, typename [type\\_traits::value\\_type](#), typename [type\\_traits::pointer](#), typename [type\\_traits::const\\_pointer](#), typename [type\\_traits::reference](#), typename [type\\_traits↵::const\\_reference](#), false, \_Alloc > **reverse\_iterator**

## 4.988.1 Detailed Description

```
template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn, typename _Alloc_ > class
Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
```

Specialization.

Definition at line 81 of file [splay\\_tree\\_/traits.hpp](#).

## 4.988.2 Member Typedef Documentation

## 4.988.2.1 node\_const\_iterator

```
typedef bin_search_tree_const_node_it < splay_tree_node < types_traits < Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >↵
::type, _Alloc > , point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits <
Key, null_type , Cmp_Fn, Node_Update, splay_tree_node < types_traits < Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch < Key, null_type, Cmp_Fn, Node_Update, _Alloc >↵
::type, _Alloc > , _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

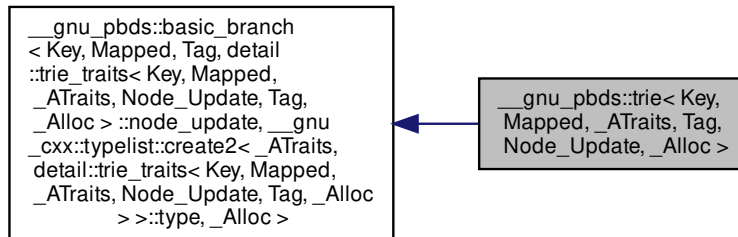
Definition at line 128 of file [bin\\_search\\_tree\\_/traits.hpp](#).

The documentation for this struct was generated from the following file:

- [splay\\_tree\\_/traits.hpp](#)

#### 4.989 `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >`:



## Public Types

- typedef `_ATraits` `access_traits`
- typedef `detail::trie_traits` < Key, Mapped, `_ATraits`, Node\_Update, Tag, `_Alloc` > ::node\_update **node\_update**

## Public Member Functions

- **trie** (const **access\_traits** &)
- template<typename It >  
**trie** (It first, It last)
- template<typename It >  
**trie** (It first, It last, const **access\_traits** &)
- **trie** (const **trie** &other)
- **trie** & **operator=** (const **trie** &other)
- void **swap** (**trie** &other)

#### 4.989.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename
Tag = pat_trie_tag, template< typename Node_Cltr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update
= null_node_update, typename _Alloc = std::allocator<char>>
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >
```

A trie-based container.

## Template Parameters

|                    |                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------|
| <i>Key</i>         | Key type.                                                                                                                |
| <i>Mapped</i>      | Map type.                                                                                                                |
| <i>ATraits</i>     | Element access traits.                                                                                                   |
| <i>Tag</i>         | Instantiating data structure type, see container_tag.                                                                    |
| <i>Node_Update</i> | Updates trie internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants. |

Base tag choice is pat\_trie\_tag.

Base is basic\_branch.

Definition at line 731 of file assoc\_container.hpp.

## 4.989.2 Member Typedef Documentation

### 4.989.2.1 access\_traits

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename
_Alloc = std::allocator<char>>>
typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_traits
```

Element access traits type.

Definition at line 738 of file assoc\_container.hpp.

## 4.989.3 Constructor & Destructor Documentation

### 4.989.3.1 trie() [1/3]

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename
_Alloc = std::allocator<char>>>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 const access_traits & t) [inline]
```

Constructor taking some policy objects. r\_access\_traits will be copied by the \_ATraits object of the container object.

Definition at line 744 of file assoc\_container.hpp.



4.989.3.2 `trie()` [2/3]

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename
_Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 It first,
 It last) [inline]
```

Constructor taking `__iterators` to a range of `value_types`. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 751 of file `assoc_container.hpp`.

4.989.3.3 `trie()` [3/3]

```
template<typename Key , typename Mapped , typename _ATraits = typename detail::default_trie_↵
access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename
Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename
_Alloc = std::allocator<char>>>
template<typename It >
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
 It first,
 It last,
 const access_traits & t) [inline]
```

Constructor taking `__iterators` to a range of `value_types` and some policy objects. The `value_types` between `first_it` and `last_it` will be inserted into the container object.

Definition at line 758 of file `assoc_container.hpp`.

The documentation for this class was generated from the following file:

- [assoc\\_container.hpp](#)

4.990 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >` Struct Template Reference

## 4.990.1 Detailed Description

```
template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
```

Trie metadata helper.

Definition at line 58 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.991 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >` Struct Template Reference

##### Public Types

- `typedef Node_Update::metadata_type type`

##### 4.991.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
```

Specialization, false.

Definition at line 62 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.992 `__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >` Struct Template Reference

##### Public Types

- `typedef null\_type type`

##### 4.992.1 Detailed Description

```
template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
```

Specialization, true.

Definition at line 69 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.993 `__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >` Struct Template Reference

##### Public Types

- `typedef trie\_metadata\_helper< __node_u, null_update >::type type`

#### 4.993.1 Detailed Description

```
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_Cltr, typename Const_Iterator, typename Cmp_Fn, typename _Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
```

Trie node metadata dispatch.

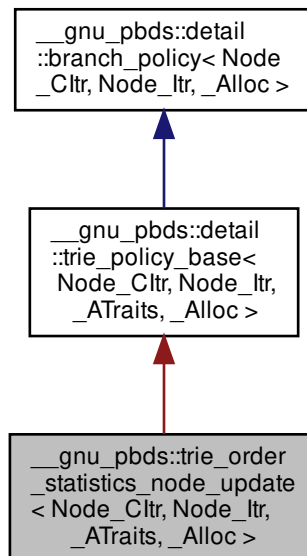
Definition at line 84 of file `trie_policy/node_metadata_selector.hpp`.

The documentation for this struct was generated from the following file:

- [trie\\_policy/node\\_metadata\\_selector.hpp](#)

#### 4.994 `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >` Class Template Reference

Inheritance diagram for `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`:



### Public Types

- typedef access\_traits::const\_iterator **a\_const\_iterator**
- typedef \_ATraits **access\_traits**
- typedef \_Alloc **allocator\_type**
- typedef node\_const\_iterator::value\_type **const\_iterator**
- typedef node\_iterator::value\_type **iterator**
- typedef base\_type::key\_const\_reference **key\_const\_reference**
- typedef base\_type::key\_type **key\_type**
- typedef size\_type **metadata\_type**
- typedef Node\_Cltr **node\_const\_iterator**
- typedef Node\_Itr **node\_iterator**
- typedef allocator\_type::size\_type **size\_type**

### Public Member Functions

- const\_iterator [find\\_by\\_order](#) (size\_type) const
- iterator [find\\_by\\_order](#) (size\_type)
- size\_type [order\\_of\\_key](#) (key\_const\_reference) const
- size\_type [order\\_of\\_prefix](#) (a\_const\_iterator, a\_const\_iterator) const

### Protected Member Functions

- void [operator\(\)](#) (node\_iterator, node\_const\_iterator) const

### Private Types

- typedef Node\_Itr::value\_type **it\_type**
- typedef remove\_const< key\_type >::type **rckey\_type**
- typedef remove\_const< value\_type >::type **rcvalue\_type**
- typedef rebind\_traits< \_Alloc, rckey\_type > **rebind\_k**
- typedef rebind\_traits< \_Alloc, rcvalue\_type > **rebind\_v**
- typedef rebind\_v::reference **reference**
- typedef [std::iterator\\_traits](#)< it\_type >::value\_type **value\_type**

### Private Member Functions

- virtual const\_iterator **end** () const =0
- it\_type **end\_iterator** () const
- virtual const access\_traits & **get\_access\_traits** () const =0

### Static Private Member Functions

- static size\_type **common\_prefix\_len** (node\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static key\_const\_reference **extract\_key** (const\_reference r\_val)
- static iterator **leftmost\_it** (node\_iterator)
- static bool **less** (e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, e\_const\_iterator, const access\_traits &)
- static iterator **rightmost\_it** (node\_iterator)

#### 4.994.1 Detailed Description

```
template<typename Node_CIttr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >
```

Functor updating ranks of entrees.

Definition at line 253 of file `trie_policy.hpp`.

#### 4.994.2 Member Function Documentation

##### 4.994.2.1 `find_by_order()` [1/2]

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
 size_type order) const [inline]
```

Finds an entry by `__order`. Returns a `const_iterator` to the entry with the `__order` order, or a `const_iterator` to the container object's end if order is at least the size of the container object.

Definition at line 81 of file `trie_policy.hpp`.

##### 4.994.2.2 `find_by_order()` [2/2]

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_stat
Node_CIttr, Node_Itr, _ATraits, _Alloc >::find_by_order (
 size_type order) [inline]
```

Finds an entry by `__order`. Returns an iterator to the entry with the `__order` order, or an iterator to the container object's end if order is at least the size of the container object.

Definition at line 47 of file `trie_policy.hpp`.

##### 4.994.2.3 `operator()()`

```
template<typename Node_CIttr , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::trie_order_statistics_node_update< Node_CIttr, Node_Itr, _ATraits, _Alloc >↵
::operator() (
 node_iterator nd_it,
 node_const_iterator) const [inline], [protected]
```

Updates the rank of a node through a `node_iterator` `node_it`; `end_nd_it` is the end node iterator.

Definition at line 154 of file `trie_policy.hpp`.

#### 4.994.2.4 order\_of\_key()

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_key (
 key_const_reference r_key) const [inline]
```

Returns the order of a key within a sequence. For example, if `r_key` is the smallest key, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

Definition at line 87 of file `trie_policy.hpp`.

#### 4.994.2.5 order\_of\_prefix()

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
 a_const_iterator b,
 a_const_iterator e) const [inline]
```

Returns the order of a prefix within a sequence. For example, if `[b, e]` is the smallest prefix, this method will return 0; if `r_key` is a key between the smallest and next key, this method will return 1; if `r_key` is a key larger than the largest key, this method will return the size of `r_c`.

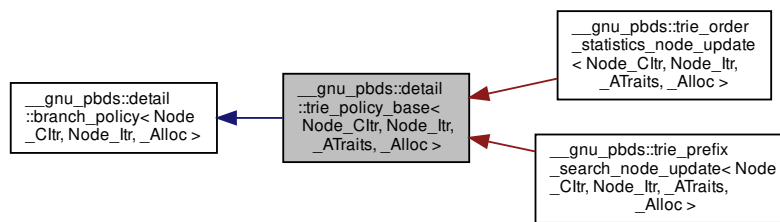
Definition at line 98 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

### 4.995 \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_CItr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference

Inheritance diagram for `__gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >`:



## Public Types

- typedef `_ATraits` **access\_traits**
- typedef `_Alloc` **allocator\_type**
- typedef `node_const_iterator::value_type` **const\_iterator**
- typedef `node_iterator::value_type` **iterator**
- typedef `base_type::key_const_reference` **key\_const\_reference**
- typedef `base_type::key_type` **key\_type**
- typedef `null_type` **metadata\_type**
- typedef `Node_Cltr` **node\_const\_iterator**
- typedef `Node_Itr` **node\_iterator**
- typedef `allocator_type::size_type` **size\_type**

## Protected Types

- typedef `rebind_v::const_pointer` **const\_pointer**
- typedef `rebind_v::const_reference` **const\_reference**
- typedef `Node_Itr::value_type` **it\_type**
- typedef `remove_const< key_type >::type` **rkey\_type**
- typedef `remove_const< value_type >::type` **rcvalue\_type**
- typedef `rebind_traits< _Alloc, rkey_type >` **rebind\_k**
- typedef `rebind_traits< _Alloc, rcvalue_type >` **rebind\_v**
- typedef `rebind_v::reference` **reference**
- typedef `std::iterator_traits< it_type >::value_type` **value\_type**

## Protected Member Functions

- virtual `const_iterator` **end** () const =0
- virtual `iterator` **end** ()=0
- `it_type` **end\_iterator** () const
- virtual `const access_traits &` **get\_access\_traits** () const =0
- virtual `node_const_iterator` **node\_begin** () const =0
- virtual `node_iterator` **node\_begin** ()=0
- virtual `node_const_iterator` **node\_end** () const =0
- virtual `node_iterator` **node\_end** ()=0

## Static Protected Member Functions

- static `size_type` **common\_prefix\_len** (`node_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `key_const_reference` **extract\_key** (`const_reference r_val`)
- static `iterator` **leftmost\_it** (`node_iterator`)
- static `bool` **less** (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `const access_traits &`)
- static `iterator` **rightmost\_it** (`node_iterator`)

#### 4.995.1 Detailed Description

```
template<typename Node_Cltr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::detail::trie_policy_base< Node_Cltr, Node_Itr, _ATraits, _Alloc >
```

Base class for trie policies.

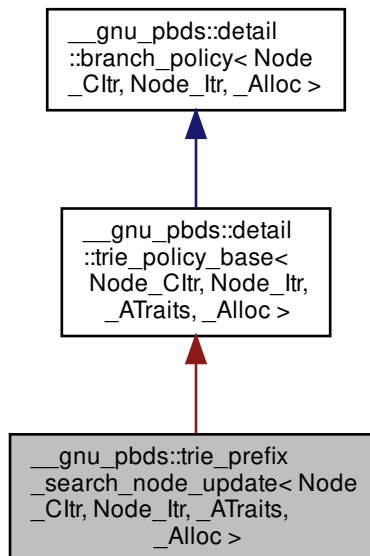
Definition at line 53 of file trie\_policy\_base.hpp.

The documentation for this class was generated from the following file:

- [trie\\_policy\\_base.hpp](#)

#### 4.996 \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc > Class Template Reference

Inheritance diagram for \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >:





## Public Types

- typedef `access_traits::const_iterator` `a_const_iterator`
- typedef `_ATraits` `access_traits`
- typedef `_Alloc` `allocator_type`
- typedef `node_const_iterator::value_type` `const_iterator`
- typedef `node_iterator::value_type` `iterator`
- typedef `base_type::key_const_reference` `key_const_reference`
- typedef `base_type::key_type` `key_type`
- typedef `null_type` `metadata_type`
- typedef `Node_Cltr` `node_const_iterator`
- typedef `Node_Itr` `node_iterator`
- typedef `allocator_type::size_type` `size_type`

## Public Member Functions

- `std::pair< const_iterator, const_iterator >` `prefix_range` (`key_const_reference`) const
- `std::pair< iterator, iterator >` `prefix_range` (`key_const_reference`)
- `std::pair< const_iterator, const_iterator >` `prefix_range` (`a_const_iterator`, `a_const_iterator`) const
- `std::pair< iterator, iterator >` `prefix_range` (`a_const_iterator`, `a_const_iterator`)

## Protected Member Functions

- void `operator()` (`node_iterator` `node_it`, `node_const_iterator` `end_nd_it`) const

## Private Types

- typedef `rebind_v::const_pointer` `const_pointer`
- typedef `rebind_v::const_reference` `const_reference`
- typedef `Node_Itr::value_type` `it_type`
- typedef `remove_const< key_type >::type` `rckey_type`
- typedef `remove_const< value_type >::type` `rcvalue_type`
- typedef `rebind_traits< _Alloc, rckey_type >` `rebind_k`
- typedef `rebind_traits< _Alloc, rcvalue_type >` `rebind_v`
- typedef `rebind_v::reference` `reference`
- typedef `std::iterator_traits< it_type >::value_type` `value_type`

## Private Member Functions

- `it_type` `end_iterator` () const

## Static Private Member Functions

- static `size_type` `common_prefix_len` (`node_iterator`, `e_const_iterator`, `e_const_iterator`, const `access_traits` &)
- static `key_const_reference` `extract_key` (`const_reference` `r_val`)
- static `iterator` `leftmost_it` (`node_iterator`)
- static bool `less` (`e_const_iterator`, `e_const_iterator`, `e_const_iterator`, `e_const_iterator`, const `access_traits` &)
- static `iterator` `rightmost_it` (`node_iterator`)

#### 4.996.1 Detailed Description

```
template<typename Node_Citr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_Citr, Node_Itr, _ATraits, _Alloc >
```

A node updator that allows tries to be searched for the range of values that match a certain prefix.

Definition at line 155 of file trie\_policy.hpp.

#### 4.996.2 Member Typedef Documentation

##### 4.996.2.1 a\_const\_iterator

```
template<typename Node_Citr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_Citr,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
```

Const element iterator.

Definition at line 168 of file trie\_policy.hpp.

##### 4.996.2.2 access\_traits

```
template<typename Node_Citr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_Citr, Node_Itr, _ATraits, _
Alloc >::access_traits
```

Element access traits.

Definition at line 165 of file trie\_policy.hpp.

##### 4.996.2.3 allocator\_type

```
template<typename Node_Citr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_Citr, Node_Itr, _ATraits, _Alloc
>::allocator_type
```

\_Alloc type.

Definition at line 171 of file trie\_policy.hpp.

#### 4.996.2.4 `size_type`

```
template<typename Node_CIter, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::size_type
```

Size type.

Definition at line 174 of file `trie_policy.hpp`.

### 4.996.3 Member Function Documentation

#### 4.996.3.1 `operator()`

```
template<typename Node_CIter , typename Node_Itr , typename _ATraits , typename _Alloc >
void __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::operator()
(
 node_iterator node_it,
 node_const_iterator end_nd_it) const [inline], [protected]
```

Called to update a node's metadata.

Definition at line 141 of file `trie_policy.hpp`.

#### 4.996.3.2 `prefix_range()` [1/4]

```
template<typename Node_CIter , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::const_iterator,
typename trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::prefix_range (
 key_const_reference r_key) const
```

Finds the const iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 49 of file `trie_policy.hpp`.

#### 4.996.3.3 `prefix_range()` [2/4]

```
template<typename Node_CIter , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::iterator,
typename trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CIter, Node_Itr, _ATraits, _Alloc >::prefix_range (
 key_const_reference r_key)
```

Finds the iterator range corresponding to all values whose prefixes match `r_key`.

Definition at line 60 of file `trie_policy.hpp`.

4.996.3.4 `prefix_range()` [3/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↵
_Alloc >::prefix_range (
 a_const_iterator b,
 a_const_iterator e) const
```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 71 of file `trie_policy.hpp`.

4.996.3.5 `prefix_range()` [4/4]

```
template<typename Node_CItr , typename Node_Itr , typename _ATraits , typename _Alloc >
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↵
::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
 a_const_iterator b,
 a_const_iterator e)
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

Definition at line 86 of file `trie_policy.hpp`.

The documentation for this class was generated from the following file:

- [trie\\_policy.hpp](#)

4.997 `__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >` Struct Template Reference

## Public Types

- enum { **reverse** }
- enum { **min\_e\_val**, **max\_e\_val**, **max\_size** }
- typedef detail::\_\_conditional\_type< Reverse, typename String::const\_reverse\_iterator, typename String::const↵\_iterator >::\_\_type **const\_iterator**
- typedef std::iterator\_traits< **const\_iterator** >::value\_type **e\_type**
- typedef detail::rebind\_traits< \_Alloc, key\_type >::const\_reference **key\_const\_reference**
- typedef String **key\_type**
- typedef \_Alloc::size\_type **size\_type**

## Static Public Member Functions

- static `const_iterator begin` (key\_const\_reference)
- static `size_type e_pos` (e\_type e)
- static `const_iterator end` (key\_const\_reference)

## 4.997.1 Detailed Description

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>>
struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >
```

Element access traits for string types.

## Template Parameters

|                  |                                                   |
|------------------|---------------------------------------------------|
| <i>String</i>    | String type.                                      |
| <i>Min_E_Val</i> | Minimal element value.                            |
| <i>Max_E_Val</i> | Maximum element value.                            |
| <i>Reverse</i>   | Reverse iteration should be used. Default: false. |
| <i>_Alloc</i>    | Allocator type.                                   |

Definition at line 74 of file `trie_policy.hpp`.

## 4.997.2 Member Typedef Documentation

4.997.2.1 `const_iterator`

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>>
typedef detail::__conditional_type<Reverse, typename String::const_reverse_iterator, typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator
```

Element const iterator type.

Definition at line 90 of file `trie_policy.hpp`.

#### 4.997.2.2 e\_type

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__↵
numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail↵
::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc =
std::allocator<char>>
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

Definition at line 93 of file trie\_policy.hpp.

#### 4.997.3 Member Function Documentation

##### 4.997.3.1 begin()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (
 key_const_reference r_key) [inline], [static]
```

Returns a const\_iterator to the first element of key\_const\_reference agumnet.

Definition at line 59 of file trie\_policy.hpp.

##### 4.997.3.2 e\_pos()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
 e_type e) [inline], [static]
```

Maps an element to a position.

Definition at line 51 of file trie\_policy.hpp.

## 4.997.3.3 end()

```
template<typename String , typename String::value_type Min_E_Val, typename String::value_type
Max_E_Val, bool Reverse, typename _Alloc >
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits<
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
 key_const_reference r_key) [inline], [static]
```

Returns a const\_iterator to the after-last element of key\_const\_reference argument.

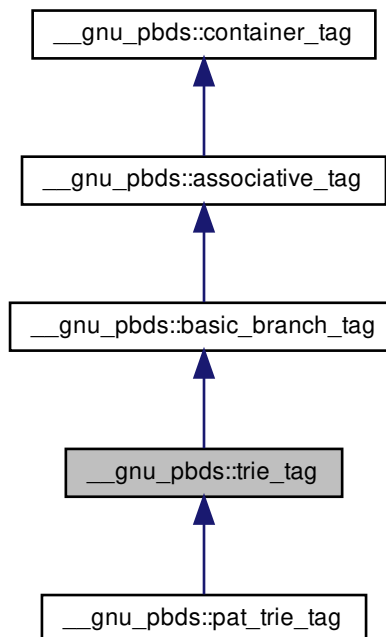
Definition at line 67 of file trie\_policy.hpp.

The documentation for this struct was generated from the following file:

- [trie\\_policy.hpp](#)

## 4.998 \_\_gnu\_pbds::trie\_tag Struct Reference

Inheritance diagram for \_\_gnu\_pbds::trie\_tag:



#### 4.998.1 Detailed Description

Basic trie structure.

Definition at line 162 of file tag\_and\_trait.hpp.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.999 `__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >` Struct Template Reference

#### 4.999.1 Detailed Description

```
template<typename Key, typename Data, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename _ATraits←
_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >
```

Trie traits class, primary template.

Definition at line 83 of file branch\_policy/traits.hpp.

The documentation for this struct was generated from the following file:

- [branch\\_policy/traits.hpp](#)

### 4.1000 `__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

#### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `base_type::_Iter< node, leaf, head, inode, true >` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `base_type::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > *` **null\_node\_update\_pointer**
- typedef `base_type::_Iter< node, leaf, head, inode, false >` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits >` **synth\_access\_traits**



#### 4.1000.1 Detailed Description

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CIttr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 62 of file `pat_trie_/traits.hpp`.

#### 4.1000.2 Member Typedef Documentation

##### 4.1000.2.1 `node_const_iterator`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CIttr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 88 of file `pat_trie_/traits.hpp`.

##### 4.1000.2.2 `node_update`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CIttr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

Definition at line 93 of file `pat_trie_/traits.hpp`.

##### 4.1000.2.3 `synth_access_traits`

```
template<typename Key , typename Mapped , typename _ATraits , template< typename Node_CIttr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

Definition at line 74 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

## 4.1001 `__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >` Struct Template Reference

### Public Types

- typedef `_ATraits` **access\_traits**
- typedef `base_type::_Cltr< node, leaf, head, inode, true >` **const\_iterator**
- typedef `base_type::_Cltr< node, leaf, head, inode, false >` **const\_reverse\_iterator**
- typedef `base_type::_Head< synth_access_traits, metadata >` **head**
- typedef `base_type::_Inode< synth_access_traits, metadata >` **inode**
- typedef `const_iterator` **iterator**
- typedef `base_type::_Leaf< synth_access_traits, metadata >` **leaf**
- typedef `base_type::_Metadata< metadata_type, _Alloc >` **metadata**
- typedef `trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type` **metadata\_type**
- typedef `base_type::_Node_base< synth_access_traits, metadata >` **node**
- typedef `base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >` **node\_const\_iterator**
- typedef `node_const_iterator` **node\_iterator**
- typedef `Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc >` **node\_update**
- typedef `null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * null_node_update`↵  
**\_pointer**
- typedef `const_reverse_iterator` **reverse\_iterator**
- typedef `__gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits >` **synth\_access\_traits**

### 4.1001.1 Detailed Description

```
template<typename Key, typename _ATraits, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _↵
Alloc > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
```

Specialization.

Definition at line 109 of file `pat_trie_/traits.hpp`.

### 4.1001.2 Member Typedef Documentation

#### 4.1001.2.1 `node_const_iterator`

```
template<typename Key , typename _ATraits , template< typename Node_Cltr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tr
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

Definition at line 135 of file `pat_trie_/traits.hpp`.

#### 4.1001.2.2 `node_update`

```
template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

Definition at line 140 of file `pat_trie_/traits.hpp`.

#### 4.1001.2.3 `synth_access_traits`

```
template<typename Key , typename _ATraits , template< typename Node_CItr, typename Node_Itr,
typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc >
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

Definition at line 121 of file `pat_trie_/traits.hpp`.

The documentation for this struct was generated from the following file:

- [pat\\_trie\\_/traits.hpp](#)

### 4.1002 `__gnu_pbds::trivial_iterator_tag` Struct Reference

#### 4.1002.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of `std::iterators`'s movement abilities.

Definition at line 75 of file `tag_and_trait.hpp`.

The documentation for this struct was generated from the following file:

- [tag\\_and\\_trait.hpp](#)

### 4.1003 `std::try_to_lock_t` Struct Reference

#### 4.1003.1 Detailed Description

Try to acquire ownership of the mutex without blocking.

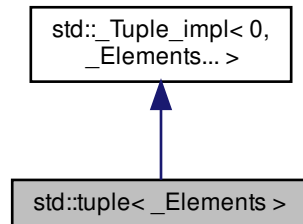
Definition at line 132 of file `std_mutex.h`.

The documentation for this struct was generated from the following file:

- [std\\_mutex.h](#)

## 4.1004 std::tuple< \_Elements > Class Template Reference

Inheritance diagram for std::tuple< \_Elements >:



### Public Member Functions

- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>`  
`constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `template<bool _NotEmpty = (sizeof...( _Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>`  
`constexpr tuple (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())`
- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements... > = true>`  
`constexpr tuple (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())`
- `template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements... > = false>`  
`constexpr tuple (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())`
- `constexpr tuple (const tuple &)=default`
- `constexpr tuple (tuple &&)=default`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !__use_other_ctor<const tuple<_<←< _UElements...>>>(), _ImplicitCtor< _Valid, const _UElements &... > = true>`  
`constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !__use_other_ctor<const tuple<_<←< _UElements...>>>(), _ExplicitCtor< _Valid, const _UElements &... > = false>`  
`constexpr tuple (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !__use_other_ctor<tuple<_U<←< _Elements...>>>(), _ImplicitCtor< _Valid, _UElements... > = true>`  
`constexpr tuple (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())`
- `template<typename... _UElements, bool _Valid = (sizeof...( _Elements) == sizeof...( _UElements)) && !__use_other_ctor<tuple<_U<←< _Elements...>>>(), _ExplicitCtor< _Valid, _UElements... > = false>`  
`constexpr tuple (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())`
- `template<typename _Alloc , _ImplicitDefaultCtor< is_object< _Alloc >::value > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a)`
- `template<typename _Alloc , bool _NotEmpty = (sizeof...( _Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>`  
`constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)`

- `template<typename _Alloc, bool _NotEmpty = (sizeof...(_Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements...  
> = true>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements...  
> = false>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)`
- `template<typename _Alloc >  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)`
- `template<typename _Alloc >  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !_use_other_↵  
ctor<const tuple<_UElements...>&>(), _ImplicitCtor< _Valid, const _UElements &... > = true>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !_use_other_↵  
ctor<const tuple<_UElements...>&>(), _ExplicitCtor< _Valid, const _UElements &... > = false>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !_use_other_↵  
ctor<tuple<_UElements...>&&(), _ImplicitCtor< _Valid, _UElements... > = true>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !_use_other_↵  
ctor<tuple<_UElements...>&&(), _ExplicitCtor< _Valid, _UElements... > = false>  
constexpr tuple (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)`
- `constexpr tuple & operator= (typename conditional< __assignable< const _Elements &... >(), const tuple &,  
const __nonesuch & >::type __in) noexcept(__nothrow_assignable< const _Elements &... >())`
- `constexpr tuple & operator= (typename conditional< __assignable< _Elements... >(), tuple &&, __nonesuch  
&& >::type __in) noexcept(__nothrow_assignable< _Elements... >())`
- `template<typename... _UElements>  
constexpr __enable_if_t< __assignable< const _UElements &... >(), tuple & > operator= (const tuple< _U↵  
Elements... > &__in) noexcept(__nothrow_assignable< const _UElements &... >())`
- `template<typename... _UElements>  
constexpr __enable_if_t< __assignable< _UElements... >(), tuple & > operator= (tuple< _UElements... >  
&&__in) noexcept(__nothrow_assignable< _UElements... >())`
- `constexpr void swap (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _Elements >... >::value)`

#### 4.1004.1 Detailed Description

```
template<typename... _Elements>
class std::tuple< _Elements >
```

Primary class template, tuple.

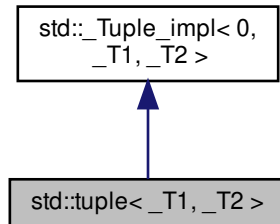
Definition at line 57 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

## 4.1005 std::tuple<\_T1, \_T2> Class Template Reference

Inheritance diagram for std::tuple<\_T1, \_T2>:



### Public Member Functions

- `template<bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>`  
`constexpr tuple (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = true>`  
`constexpr tuple (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ExplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = false>`  
`constexpr tuple (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `constexpr tuple (const tuple &)=default`
- `constexpr tuple (tuple &&)=default`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>`  
`constexpr tuple (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>`  
`constexpr tuple (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())`
- `template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>`  
`constexpr tuple (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())`

- template<typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (pair< \_U1, \_U2 > && \_\_in) noexcept(\_\_nothrow\_constructible< \_U1, \_U2 >())
- template<typename \_Alloc, \_ImplicitDefaultCtor< is\_object< \_Alloc >::value, \_T1, \_T2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a)
- template<typename \_Alloc, bool \_Dummy = true, \_ImplicitCtor< \_Dummy, const \_T1 &, const \_T2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const \_T1 & \_\_a1, const \_T2 & \_\_a2)
- template<typename \_Alloc, bool \_Dummy = true, \_ExplicitCtor< \_Dummy, const \_T1 &, const \_T2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const \_T1 & \_\_a1, const \_T2 & \_\_a2)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, \_U1, \_U2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, \_U1 && \_\_a1, \_U2 && \_\_a2)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, \_U1 && \_\_a1, \_U2 && \_\_a2)
- template<typename \_Alloc >  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple & \_\_in)
- template<typename \_Alloc >  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, tuple && \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, const \_U1 &, const \_U2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, const \_U1 &, const \_U2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const tuple< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, \_U1, \_U2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, tuple< \_U1, \_U2 > && \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, tuple< \_U1, \_U2 > && \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, const \_U1 &, const \_U2 & > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const pair< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, const \_U1 &, const \_U2 & > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, const pair< \_U1, \_U2 > & \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ImplicitCtor< true, \_U1, \_U2 > = true>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, pair< \_U1, \_U2 > && \_\_in)
- template<typename \_Alloc, typename \_U1, typename \_U2, \_ExplicitCtor< true, \_U1, \_U2 > = false>  
constexpr tuple (allocator\_arg\_t \_\_tag, const \_Alloc & \_\_a, pair< \_U1, \_U2 > && \_\_in)
- constexpr tuple & operator= (typename conditional< \_\_assignable< const \_T1 &, const \_T2 & >(), const tuple  
&, const \_\_nonesuch & >::type \_\_in) noexcept(\_\_nothrow\_assignable< const \_T1 &, const \_T2 & >())
- constexpr tuple & operator= (typename conditional< \_\_assignable< \_T1, \_T2 >(), tuple &&, \_\_nonesuch &&  
>::type \_\_in) noexcept(\_\_nothrow\_assignable< \_T1, \_T2 >())
- template<typename \_U1, typename \_U2 >  
constexpr \_\_enable\_if\_t< \_\_assignable< const \_U1 &, const \_U2 & >(), tuple & > operator= (const tuple< \_U1,  
\_U2 > & \_\_in) noexcept(\_\_nothrow\_assignable< const \_U1 &, const \_U2 & >())
- template<typename \_U1, typename \_U2 >  
constexpr \_\_enable\_if\_t< \_\_assignable< \_U1, \_U2 >(), tuple & > operator= (tuple< \_U1, \_U2 > && \_\_in)  
noexcept(\_\_nothrow\_assignable< \_U1, \_U2 >())
- template<typename \_U1, typename \_U2 >  
constexpr \_\_enable\_if\_t< \_\_assignable< const \_U1 &, const \_U2 & >(), tuple & > operator= (const pair< \_U1,  
\_U2 > & \_\_in) noexcept(\_\_nothrow\_assignable< const \_U1 &, const \_U2 & >())
- template<typename \_U1, typename \_U2 >  
constexpr \_\_enable\_if\_t< \_\_assignable< \_U1, \_U2 >(), tuple & > operator= (pair< \_U1, \_U2 > && \_\_in)  
noexcept(\_\_nothrow\_assignable< \_U1, \_U2 >())
- constexpr void swap (tuple & \_\_in) noexcept(\_\_and< \_\_is\_nothrow\_swappable< \_T1 >, \_\_is\_nothrow\_swappable< \_T2 > >::value)

#### 4.1005.1 Detailed Description

```
template<typename _T1, typename _T2>
class std::tuple< _T1, _T2 >
```

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.

Definition at line 891 of file tuple.

The documentation for this class was generated from the following file:

- [tuple](#)

#### 4.1006 std::tuple\_element< \_Int, \_Tp > Struct Template Reference

##### 4.1006.1 Detailed Description

```
template<std::size_t _Int, typename _Tp>
struct std::tuple_element< _Int, _Tp >
```

tuple\_element

Gives the type of the ith element of a given tuple type.

Definition at line 428 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

#### 4.1007 std::tuple\_element< 0, std::pair< \_Tp1, \_Tp2 > > Struct Template Reference

##### Public Types

- typedef \_Tp1 **type**

##### 4.1007.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 0, std::pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.

Definition at line 162 of file utility.

The documentation for this struct was generated from the following file:

- [utility](#)



## 4.1008 `std::tuple_element< 0, tuple< _Head, _Tail... > >` Struct Template Reference

### Public Types

- `typedef _Head type`

#### 4.1008.1 Detailed Description

```
template<typename _Head, typename... _Tail>
struct std::tuple_element< 0, tuple< _Head, _Tail... > >
```

Basis case for `tuple_element`: The first element is the one we're seeking.

Definition at line 1270 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1009 `std::tuple_element< 1, std::pair< _Tp1, _Tp2 > >` Struct Template Reference

### Public Types

- `typedef _Tp2 type`

#### 4.1009.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_element< 1, std::pair< _Tp1, _Tp2 > >
```

Partial specialization for `std::pair`.

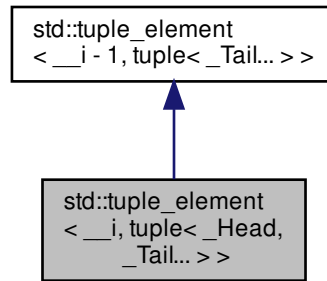
Definition at line 167 of file `utility`.

The documentation for this struct was generated from the following file:

- [utility](#)

#### 4.1010 `std::tuple_element< __i, tuple< _Head, _Tail... > >` Struct Template Reference

Inheritance diagram for `std::tuple_element< __i, tuple< _Head, _Tail... > >`:



##### 4.1010.1 Detailed Description

```
template<std::size_t __i, typename _Head, typename... _Tail>
struct std::tuple_element< __i, tuple< _Head, _Tail... > >
```

Recursive case for `tuple_element`: strip off the first element in the tuple and retrieve the (i-1)th element of the remaining tuple.

Definition at line 1263 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

#### 4.1011 `std::tuple_element< __i, tuple<> >` Struct Template Reference

##### 4.1011.1 Detailed Description

```
template<size_t __i>
struct std::tuple_element< __i, tuple<> >
```

Error case for `tuple_element`: invalid index.

Definition at line 1279 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1012 `std::tuple_element<_Int, ::array<_Tp, _Nm>>` Struct Template Reference

### Public Types

- `typedef _Tp type`

#### 4.1012.1 Detailed Description

```
template<std::size_t _Int, typename _Tp, std::size_t _Nm>
struct std::tuple_element<_Int, ::array<_Tp, _Nm>>
```

Partial specialization for `std::array`.

Definition at line 432 of file `array`.

The documentation for this struct was generated from the following file:

- [array](#)

## 4.1013 `std::tuple_element<_Int, std::__debug::array<_Tp, _Nm>>` Struct Template Reference

### Public Types

- `typedef _Tp type`

#### 4.1013.1 Detailed Description

```
template<std::size_t _Int, typename _Tp, std::size_t _Nm>
struct std::tuple_element<_Int, std::__debug::array<_Tp, _Nm>>
```

`tuple_element`

Definition at line 395 of file `debug/array`.

The documentation for this struct was generated from the following file:

- [debug/array](#)

## 4.1014 `std::tuple_size<_Tp>` Struct Template Reference

Inherited by `std::tuple_size<const __enable_if_has_tuple_size<_Tp>>`, `std::tuple_size<const volatile __enable_if_has_tuple_size<_Tp>>`, and `std::tuple_size<volatile __enable_if_has_tuple_size<_Tp>>`.

#### 4.1014.1 Detailed Description

```
template<typename _Tp>
struct std::tuple_size< _Tp >
```

tuple\_size

Finds the size of a given tuple type.

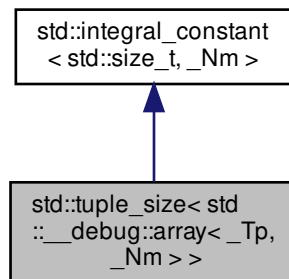
Definition at line 419 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

#### 4.1015 std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > > Struct Template Reference

Inheritance diagram for std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > >:



#### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

#### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr std::size\_t **value**

## 4.1015.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::tuple_size< std::__debug::array< _Tp, _Nm > >
```

tuple\_size

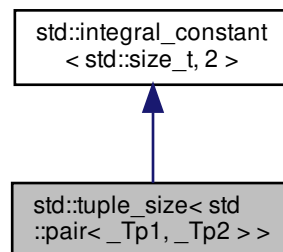
Definition at line 390 of file debug/array.

The documentation for this struct was generated from the following file:

- [debug/array](#)

## 4.1016 std::tuple\_size&lt; std::pair&lt; \_Tp1, \_Tp2 &gt; &gt; Struct Template Reference

Inheritance diagram for std::tuple\_size< std::pair< \_Tp1, \_Tp2 > >:



## Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

### Static Public Attributes

- static constexpr std::size\_t **value**

#### 4.1016.1 Detailed Description

```
template<class _Tp1, class _Tp2>
struct std::tuple_size< std::pair< _Tp1, _Tp2 > >
```

Partial specialization for std::pair.

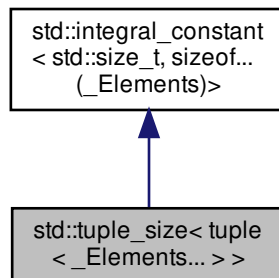
Definition at line 157 of file utility.

The documentation for this struct was generated from the following file:

- [utility](#)

#### 4.1017 std::tuple\_size< tuple< \_Elements... > > Struct Template Reference

Inheritance diagram for std::tuple\_size< tuple< \_Elements... > >:



### Public Types

- typedef [integral\\_constant](#)< std::size\_t, \_\_v > **type**
- typedef std::size\_t **value\_type**

### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

## Static Public Attributes

- static constexpr `std::size_t` **value**

## 4.1017.1 Detailed Description

```
template<typename... _Elements>
struct std::tuple_size< tuple< _Elements... > >
```

```
class tuple_size
```

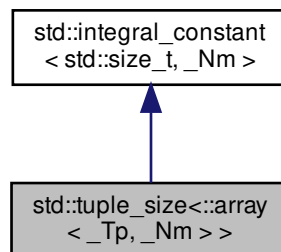
Definition at line 1250 of file `tuple`.

The documentation for this struct was generated from the following file:

- [tuple](#)

4.1018 `std::tuple_size<::array<_Tp,_Nm>>` Struct Template Reference

Inheritance diagram for `std::tuple_size<::array<_Tp,_Nm>>`:



## Public Types

- typedef [integral\\_constant](#)< `std::size_t`, `__v` > **type**
- typedef `std::size_t` **value\_type**

## Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

#### Static Public Attributes

- static constexpr std::size\_t **value**

#### 4.1018.1 Detailed Description

```
template<typename _Tp, std::size_t _Nm>
struct std::tuple_size<::array<_Tp, _Nm> > >
```

Partial specialization for std::array.

Definition at line 423 of file array.

The documentation for this struct was generated from the following file:

- [array](#)

#### 4.1019 \_\_gnu\_pbds::detail::entry\_cmp<\_VTp, Cmp\_Fn, \_Alloc, false>::type Struct Reference

Inherits Cmp\_Fn.

#### Public Member Functions

- **type** (const Cmp\_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

#### 4.1019.1 Detailed Description

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp<_VTp, Cmp_Fn, _Alloc, false>::type
```

Compare plus entry.

Definition at line 71 of file entry\_cmp.hpp.

The documentation for this struct was generated from the following file:

- [entry\\_cmp.hpp](#)



## 4.1020 std::type\_index Struct Reference

## Public Member Functions

- **type\_index** (const [type\\_info](#) &\_\_rhs) noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator<=** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator==** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>** (const [type\\_index](#) &\_\_rhs) const noexcept
- bool **operator>=** (const [type\\_index](#) &\_\_rhs) const noexcept

## 4.1020.1 Detailed Description

## Class type\_index

The class type\_index provides a simple wrapper for type\_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).

Definition at line 55 of file typeindex.

The documentation for this struct was generated from the following file:

- [typeindex](#)

## 4.1021 std::type\_info Class Reference

Inherited by [\\_\\_cxxabiv1::\\_\\_array\\_type\\_info](#), [\\_\\_cxxabiv1::\\_\\_class\\_type\\_info](#), [\\_\\_cxxabiv1::\\_\\_enum\\_type\\_info](#), [\\_\\_cxxabiv1::\\_\\_function\\_type\\_info](#), [\\_\\_cxxabiv1::\\_\\_fundamental\\_type\\_info](#), and [\\_\\_cxxabiv1::\\_\\_pbase\\_type\\_info](#).

## Public Member Functions

- virtual [~type\\_info](#) ()
- virtual bool **\_\_do\_catch** (const [type\\_info](#) \*\_\_thr\_type, void \*\*\_\_thr\_obj, unsigned \_\_outer) const
- virtual bool **\_\_do\_upcast** (const [\\_\\_cxxabiv1::\\_\\_class\\_type\\_info](#) \*\_\_target, void \*\*\_\_obj\_ptr) const
- virtual bool **\_\_is\_function\_p** () const
- virtual bool **\_\_is\_pointer\_p** () const
- bool **before** (const [type\\_info](#) &\_\_arg) const noexcept
- **size\_t hash\_code** () const noexcept
- const char \* **name** () const noexcept
- bool **operator!=** (const [type\\_info](#) &\_\_arg) const noexcept
- bool **operator==** (const [type\\_info](#) &\_\_arg) const noexcept

### Protected Member Functions

- **type\_info** (const char \* \_\_n)

### Protected Attributes

- const char \* **\_\_name**

#### 4.1021.1 Detailed Description

Part of RTTI.

The `type_info` class describes type information generated by an implementation.

Definition at line 88 of file `typeinfo`.

#### 4.1021.2 Constructor & Destructor Documentation

##### 4.1021.2.1 ~type\_info()

```
virtual std::type_info::~~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated `type_info` structures in the new-abi.

#### 4.1021.3 Member Function Documentation

##### 4.1021.3.1 name()

```
const char* std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!

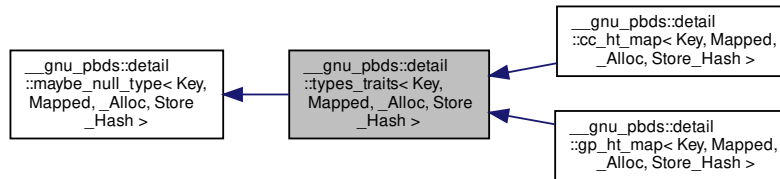
Definition at line 99 of file `typeinfo`.

The documentation for this class was generated from the following file:

- [typeinfo](#)

## 4.1022 \_\_gnu\_pbds::detail::types\_traits&lt; Key, Mapped, \_Alloc, Store\_Hash &gt; Struct Template Reference

Inheritance diagram for \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash >:



## Public Types

- typedef `std::pair< size_type, size_type >` **comp\_hash**
- typedef `__rebind_va::const_pointer` **const\_pointer**
- typedef `__rebind_va::const_reference` **const\_reference**
- typedef `__rebind_ka::const_pointer` **key\_const\_pointer**
- typedef `__rebind_ka::const_reference` **key\_const\_reference**
- typedef `__rebind_ka::pointer` **key\_pointer**
- typedef `__rebind_ka::reference` **key\_reference**
- typedef `Key` **key\_type**
- typedef `__rebind_ma::const_pointer` **mapped\_const\_pointer**
- typedef `__rebind_ma::const_reference` **mapped\_const\_reference**
- typedef `__rebind_ma::pointer` **mapped\_pointer**
- typedef `__rebind_ma::reference` **mapped\_reference**
- typedef `Mapped` **mapped\_type**
- typedef `__nothrowcopy::indicator` **no\_throw\_indicator**
- typedef `__rebind_va::pointer` **pointer**
- typedef `__rebind_va::reference` **reference**
- typedef `_Alloc::size_type` **size\_type**
- typedef `integral_constant< int, Store_Hash >` **store\_extra**
- typedef `stored_data< value_type, size_type, Store_Hash >` **stored\_data\_type**
- typedef `select_value_type< Key, Mapped >::type` **value\_type**

## Public Attributes

- `no_throw_indicator` **m\_no\_throw\_copies\_indicator**
- `store_extra` **m\_store\_extra\_indicator**

#### 4.1022.1 Detailed Description

```
template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >
```

Traits for abstract types.

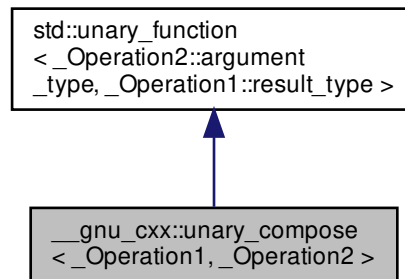
Definition at line 154 of file types\_traits.hpp.

The documentation for this struct was generated from the following file:

- [types\\_traits.hpp](#)

#### 4.1023 \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 > Class Template Reference

Inheritance diagram for \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >:



##### Public Types

- typedef \_Operation2::argument\_type [argument\\_type](#)
- typedef \_Operation1::result\_type [result\\_type](#)

##### Public Member Functions

- **unary\_compose** (const \_Operation1 &\_\_x, const \_Operation2 &\_\_y)
- \_Operation1::result\_type **operator()** (const typename \_Operation2::argument\_type &\_\_x) const

##### Protected Attributes

- \_Operation1 **\_M\_fn1**
- \_Operation2 **\_M\_fn2**

## 4.1023.1 Detailed Description

```
template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose<_Operation1, _Operation2>
```

An [SGI extension](#).

Definition at line 117 of file ext/functional.

## 4.1023.2 Member Typedef Documentation

## 4.1023.2.1 argument\_type

```
typedef _Operation2::argument_type std::unary_function<_Operation2::argument_type, _Operation1↔
::result_type>::argument_type [inherited]
```

argument\_type is the type of the argument

Definition at line 108 of file stl\_function.h.

## 4.1023.2.2 result\_type

```
typedef _Operation1::result_type std::unary_function<_Operation2::argument_type, _Operation1↔
::result_type>::result_type [inherited]
```

result\_type is the return type

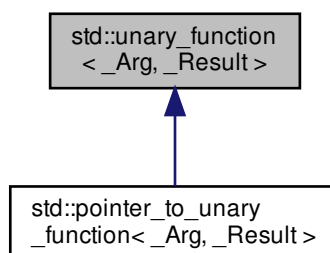
Definition at line 111 of file stl\_function.h.

The documentation for this class was generated from the following file:

- [ext/functional](#)

## 4.1024 std::unary\_function&lt;\_Arg, \_Result&gt; Struct Template Reference

Inheritance diagram for std::unary\_function<\_Arg, \_Result>:



## Public Types

- typedef `_Arg` [argument\\_type](#)
- typedef `_Result` [result\\_type](#)

### 4.1024.1 Detailed Description

```
template<typename _Arg, typename _Result>
struct std::unary_function< _Arg, _Result >
```

This is one of the [functor base classes](#).

Definition at line 105 of file `stl_function.h`.

### 4.1024.2 Member Typedef Documentation

#### 4.1024.2.1 `argument_type`

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.1024.2.2 `result_type`

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type
```

`result_type` is the return type

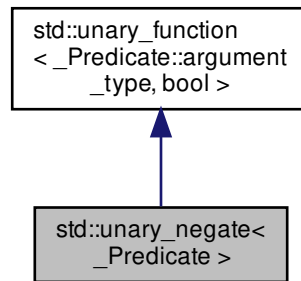
Definition at line 111 of file `stl_function.h`.

The documentation for this struct was generated from the following file:

- [stl\\_function.h](#)

## 4.1025 std::unary\_negate&lt; \_Predicate &gt; Class Template Reference

Inheritance diagram for std::unary\_negate< \_Predicate >:



## Public Types

- typedef `_Predicate::argument_type` [argument\\_type](#)
- typedef `bool` [result\\_type](#)

## Public Member Functions

- constexpr **unary\_negate** (`const _Predicate & __x`)
- constexpr `bool` **operator()** (`const typename _Predicate::argument_type & __x`) const

## Protected Attributes

- `_Predicate` **\_M\_pred**

## 4.1025.1 Detailed Description

```
template<typename _Predicate>
class std::unary_negate< _Predicate >
```

One of the [negation functors](#).

Definition at line 1003 of file `stl_function.h`.

## 4.1025.2 Member Typedef Documentation

#### 4.1025.2.1 `argument_type`

```
typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type , bool >::argument_type
[inherited]
```

`argument_type` is the type of the argument

Definition at line 108 of file `stl_function.h`.

#### 4.1025.2.2 `result_type`

```
typedef bool std::unary_function< _Predicate::argument_type , bool >::result_type [inherited]
```

`result_type` is the return type

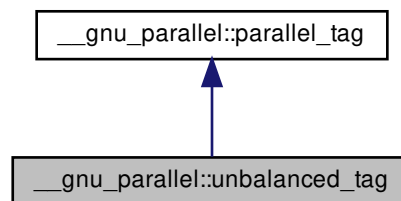
Definition at line 111 of file `stl_function.h`.

The documentation for this class was generated from the following file:

- [stl\\_function.h](#)

#### 4.1026 `__gnu_parallel::unbalanced_tag` Struct Reference

Inheritance diagram for `__gnu_parallel::unbalanced_tag`:



#### Public Member Functions

- [\\_ThreadIndex](#) `__get_num_threads` ()
- void `set_num_threads` ([\\_ThreadIndex](#) `__num_threads`)



#### 4.1026.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

Definition at line 92 of file tags.h.

#### 4.1026.2 Member Function Documentation

##### 4.1026.2.1 \_\_get\_num\_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]
```

Find out desired number of threads.

##### Returns

Desired number of threads.

Definition at line 63 of file tags.h.

Referenced by `__gnu_parallel::__parallel_sort()`.

##### 4.1026.2.2 set\_num\_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
 _ThreadIndex __num_threads) [inline], [inherited]
```

Set the desired number of threads.

##### Parameters

|                                   |                            |
|-----------------------------------|----------------------------|
| <code><i>__num_threads</i></code> | Desired number of threads. |
|-----------------------------------|----------------------------|

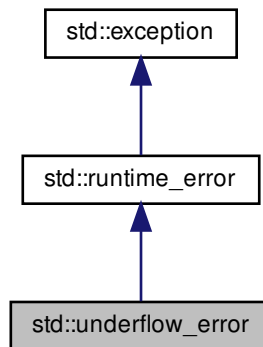
Definition at line 73 of file tags.h.

The documentation for this struct was generated from the following file:

- [tags.h](#)

## 4.1027 std::underflow\_error Class Reference

Inheritance diagram for std::underflow\_error:



### Public Member Functions

- **underflow\_error** (const [string](#) &\_\_arg) \_GLIBCXX\_TXN\_SAFE
- **underflow\_error** (const char \*) \_GLIBCXX\_TXN\_SAFE
- **underflow\_error** (const [underflow\\_error](#) &)=default
- **underflow\_error** ([underflow\\_error](#) &&)=default
- [underflow\\_error](#) & **operator=** (const [underflow\\_error](#) &)=default
- [underflow\\_error](#) & **operator=** ([underflow\\_error](#) &&)=default
- virtual const char \* [what](#) () const noexcept

### 4.1027.1 Detailed Description

Thrown to indicate arithmetic underflow.

Definition at line 288 of file stdexcept.

### 4.1027.2 Member Function Documentation

4.1027.2.1 `what()`

```
virtual const char* std::runtime_error::what () const [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from [std::exception](#).

Reimplemented in [std::experimental::filesystem::v1::filesystem\\_error](#).

The documentation for this class was generated from the following file:

- [stdexcept](#)

4.1028 `std::underlying_type< _Tp >` Struct Template Reference

Inherits `std::__underlying_type_impl< _Tp, bool >`.

## Public Types

- using **type** = `__underlying_type(_Tp)`

## 4.1028.1 Detailed Description

```
template<typename _Tp>
struct std::underlying_type< _Tp >
```

The underlying type of an enum.

Definition at line 2324 of file `type_traits`.

The documentation for this struct was generated from the following file:

- [type\\_traits](#)

4.1029 `std::uniform_int_distribution< _IntType >` Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef `_IntType` [result\\_type](#)

## Public Member Functions

- [uniform\\_int\\_distribution](#) ()
- [uniform\\_int\\_distribution](#) (\_IntType \_\_a, \_IntType \_\_b=[numeric\\_limits](#)< \_IntType >::max())
- [uniform\\_int\\_distribution](#) (const [param\\_type](#) &\_\_p)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [template](#)<typename \_UniformRandomNumberGenerator >  
void [\\_\\_generate](#) ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type a](#) () const
- [result\\_type b](#) () const
- [result\\_type max](#) () const
- [result\\_type min](#) () const
- [template](#)<typename \_UniformRandomNumberGenerator >  
[result\\_type operator](#)() (\_UniformRandomNumberGenerator &\_\_urng)
- [template](#)<typename \_UniformRandomNumberGenerator >  
[result\\_type operator](#)() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type param](#) () const
- void [param](#) (const [param\\_type](#) &\_\_param)
- void [reset](#) ()

## Friends

- bool [operator==](#) (const [uniform\\_int\\_distribution](#) &\_\_d1, const [uniform\\_int\\_distribution](#) &\_\_d2)

## 4.1029.1 Detailed Description

```
template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >
```

Uniform discrete distribution for random numbers. A discrete random distribution on the range  $[min, max]$  with equal probability throughout the range.

Definition at line 74 of file `uniform_int_dist.h`.

## 4.1029.2 Member Typedef Documentation

## 4.1029.2.1 result\_type

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

Definition at line 77 of file `uniform_int_dist.h`.

### 4.1029.3 Constructor & Destructor Documentation

#### 4.1029.3.1 `uniform_int_distribution()` [1/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution () [inline]
```

Constructs a uniform distribution object.

Definition at line 122 of file `uniform_int_dist.h`.

#### 4.1029.3.2 `uniform_int_distribution()` [2/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
 _IntType __a,
 _IntType __b = numeric_limits<_IntType>::max()) [inline], [explicit]
```

Constructs a uniform distribution object.

Definition at line 128 of file `uniform_int_dist.h`.

### 4.1029.4 Member Function Documentation

#### 4.1029.4.1 `max()`

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

Definition at line 180 of file `uniform_int_dist.h`.

#### 4.1029.4.2 `min()`

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

Definition at line 173 of file `uniform_int_dist.h`.

#### 4.1029.4.3 operator()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator >
result_type std::uniform_int_distribution< _IntType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 188 of file uniform\_int\_dist.h.

#### 4.1029.4.4 param() [1/2]

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 158 of file uniform\_int\_dist.h.

Referenced by std::operator>>().

#### 4.1029.4.5 param() [2/2]

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

##### Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 166 of file uniform\_int\_dist.h.

#### 4.1029.4.6 reset()

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the uniform integer distribution.

Definition at line 144 of file uniform\_int\_dist.h.

## 4.1029.5 Friends And Related Function Documentation

## 4.1029.5.1 operator==

```
template<typename _IntType = int>
bool operator== (
 const uniform_int_distribution< _IntType > & __d1,
 const uniform_int_distribution< _IntType > & __d2) [friend]
```

Return true if two uniform integer distributions have the same parameters.

Definition at line 223 of file uniform\_int\_dist.h.

The documentation for this class was generated from the following file:

- [uniform\\_int\\_dist.h](#)

## 4.1030 std::uniform\_real\_distribution&lt;\_RealType&gt; Class Template Reference

## Classes

- struct [param\\_type](#)

## Public Types

- typedef \_RealType [result\\_type](#)

## Public Member Functions

- [uniform\\_real\\_distribution](#) ()
- [uniform\\_real\\_distribution](#) (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **uniform\_real\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [result\\_type](#) **a** () const
- [result\\_type](#) **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator**() (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool `operator==` (const `uniform_real_distribution` &\_\_d1, const `uniform_real_distribution` &\_\_d2)

### 4.1030.1 Detailed Description

```
template<typename _RealType = double>
class std::uniform_real_distribution< _RealType >
```

Uniform continuous distribution for random numbers.

A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Definition at line 1740 of file random.h.

### 4.1030.2 Member Typedef Documentation

#### 4.1030.2.1 result\_type

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 1743 of file random.h.

### 4.1030.3 Constructor & Destructor Documentation

#### 4.1030.3.1 uniform\_real\_distribution() [1/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution () [inline]
```

Constructs a `uniform_real_distribution` object.

The lower bound is set to 0.0 and the upper bound to 1.0

Definition at line 1790 of file random.h.

#### 4.1030.3.2 uniform\_real\_distribution() [2/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
 _RealType __a,
 _RealType __b = _RealType(1)) [inline], [explicit]
```

Constructs a `uniform_real_distribution` object.



## Parameters

|                       |                                           |
|-----------------------|-------------------------------------------|
| <code>↵<br/>_a</code> | [IN] The lower bound of the distribution. |
| <code>↵<br/>_b</code> | [IN] The upper bound of the distribution. |

Definition at line 1799 of file `random.h`.

## 4.1030.4 Member Function Documentation

4.1030.4.1 `max()`

```
template<typename _RealType = double>
result_type std::uniform_real_distribution<_RealType>::max () const [inline]
```

Returns the inclusive upper bound of the distribution range.

Definition at line 1850 of file `random.h`.

4.1030.4.2 `min()`

```
template<typename _RealType = double>
result_type std::uniform_real_distribution<_RealType>::min () const [inline]
```

Returns the inclusive lower bound of the distribution range.

Definition at line 1843 of file `random.h`.

4.1030.4.3 `operator()()`

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::uniform_real_distribution<_RealType>::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 1858 of file `random.h`.

**4.1030.4.4** `param()` [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 1828 of file random.h.

Referenced by `std::operator>>()`.

**4.1030.4.5** `param()` [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

**Parameters**

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 1836 of file random.h.

**4.1030.4.6** `reset()`

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Does nothing for the uniform real distribution.

Definition at line 1814 of file random.h.

**4.1030.5** Friends And Related Function Documentation

## 4.1030.5.1 operator==

```
template<typename _RealType = double>
bool operator== (
 const uniform_real_distribution< _RealType > & __d1,
 const uniform_real_distribution< _RealType > & __d2) [friend]
```

Return true if two uniform real distributions have the same parameters.

Definition at line 1898 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

## 4.1031 std::unique\_lock&lt;\_Mutex&gt; Class Template Reference

## Public Types

- typedef \_Mutex **mutex\_type**

## Public Member Functions

- **unique\_lock** (mutex\_type &\_\_m)
- **unique\_lock** (mutex\_type &\_\_m, [defer\\_lock\\_t](#)) noexcept
- **unique\_lock** (mutex\_type &\_\_m, [try\\_to\\_lock\\_t](#))
- **unique\_lock** (mutex\_type &\_\_m, [adopt\\_lock\\_t](#)) noexcept
- template<typename \_Clock, typename \_Duration >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- template<typename \_Rep, typename \_Period >  
**unique\_lock** (mutex\_type &\_\_m, const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- **unique\_lock** (const [unique\\_lock](#) &)=delete
- **unique\_lock** ([unique\\_lock](#) &&\_\_u) noexcept
- void **lock** ()
- mutex\_type \* **mutex** () const noexcept
- **operator bool** () const noexcept
- [unique\\_lock](#) & **operator=** (const [unique\\_lock](#) &)=delete
- [unique\\_lock](#) & **operator=** ([unique\\_lock](#) &&\_\_u) noexcept
- bool **owns\_lock** () const noexcept
- mutex\_type \* **release** () noexcept
- void **swap** ([unique\\_lock](#) &\_\_u) noexcept
- bool **try\_lock** ()
- template<typename \_Rep, typename \_Period >  
bool **try\_lock\_for** (const [chrono::duration](#)< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
bool **try\_lock\_until** (const [chrono::time\\_point](#)< \_Clock, \_Duration > &\_\_atime)
- void **unlock** ()

## Related Functions

(Note that these are not member functions.)

- `template<typename _Mutex >`  
`void swap(unique_lock<_Mutex> &__x, unique_lock<_Mutex> &__y) noexcept`

### 4.1031.1 Detailed Description

```
template<typename _Mutex>
class std::unique_lock<_Mutex>
```

A movable scoped lock type.

A `unique_lock` controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another `unique_lock` by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Definition at line 56 of file `unique_lock.h`.

### 4.1031.2 Friends And Related Function Documentation

#### 4.1031.2.1 swap()

```
template<typename _Mutex >
void swap (
 unique_lock<_Mutex> & __x,
 unique_lock<_Mutex> & __y) [related]
```

Swap overload for `unique_lock` objects.

Definition at line 235 of file `unique_lock.h`.

The documentation for this class was generated from the following file:

- [unique\\_lock.h](#)

## 4.1032 std::unique\_ptr<\_Tp, \_Dp> Class Template Reference

### Public Types

- using **deleter\_type** = `_Dp`
- using **element\_type** = `_Tp`
- using **pointer** = `typename __uniq_ptr_impl<_Tp, _Dp>::pointer`

## Public Member Functions

- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
constexpr [unique\\_ptr](#) () noexcept
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
[unique\\_ptr](#) (pointer \_\_p) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_copy\_constructible<\_Del>>>  
[unique\\_ptr](#) (pointer \_\_p, const deleter\_type &\_\_d) noexcept
- template<typename \_Del = deleter\_type, typename = \_Require<is\_move\_constructible<\_Del>>>  
[unique\\_ptr](#) (pointer \_\_p, \_\_enable\_if\_t<!is\_lvalue\_reference<\_Del>::value, \_Del && > \_\_d) noexcept
- template<typename \_Del = deleter\_type, typename \_DelUnref = typename remove\_reference<\_Del>::type>  
[unique\\_ptr](#) (pointer, \_\_enable\_if\_t<is\_lvalue\_reference<\_Del>::value, \_DelUnref && >)=delete
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint<\_Del>>  
constexpr [unique\\_ptr](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) ([unique\\_ptr](#) &&)=default
- template<typename \_Up, typename >  
[unique\\_ptr](#) ([auto\\_ptr](#)<\_Up> &&\_\_u) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require<\_\_safe\_conversion\_up<\_Up, \_Ep>, typename conditional<is\_lvalue\_reference<\_Dp>::value, is\_same<\_Ep, \_Dp>, is\_convertible<\_Ep, \_Dp>>::type>>  
[unique\\_ptr](#) ([unique\\_ptr](#)<\_Up, \_Ep> &&\_\_u) noexcept
- [unique\\_ptr](#) (const [unique\\_ptr](#) &)=delete
- ~[unique\\_ptr](#) () noexcept
- pointer [get](#) () const noexcept
- deleter\_type & [get\\_deleter](#) () noexcept
- const deleter\_type & [get\\_deleter](#) () const noexcept
- [operator bool](#) () const noexcept
- [add\\_lvalue\\_reference](#)< element\_type >::type [operator\\*](#) () const
- pointer [operator->](#) () const noexcept
- [unique\\_ptr](#) & [operator=](#) ([unique\\_ptr](#) &&)=default
- template<typename \_Up, typename \_Ep >  
[enable\\_if](#)<\_\_and<\_\_safe\_conversion\_up<\_Up, \_Ep>, [is\\_assignable](#)< deleter\_type &, \_Ep && > >::value,  
[unique\\_ptr](#) & >::type [operator=](#) ([unique\\_ptr](#)<\_Up, \_Ep> &&\_\_u) noexcept
- [unique\\_ptr](#) & [operator=](#) (nullptr\_t) noexcept
- [unique\\_ptr](#) & [operator=](#) (const [unique\\_ptr](#) &)=delete
- pointer [release](#) () noexcept
- void [reset](#) (pointer \_\_p=pointer()) noexcept
- void [swap](#) ([unique\\_ptr](#) &\_\_u) noexcept

## Related Functions

(Note that these are not member functions.)

- template<typename \_Tp, typename \_Dp >  
[enable\\_if](#)<\_\_is\_swappable<\_Dp>::value >::type [swap](#) ([unique\\_ptr](#)<\_Tp, \_Dp> &\_\_x, [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_y) noexcept
- template<typename \_Tp, typename \_Dp, typename \_Up, typename \_Ep >  
bool [operator==](#) (const [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_x, const [unique\\_ptr](#)<\_Up, \_Ep> &\_\_y)
- template<typename \_Tp, typename \_Dp >  
bool [operator==](#) (const [unique\\_ptr](#)<\_Tp, \_Dp> &\_\_x, nullptr\_t) noexcept

- `template<typename _Tp, typename _Dp >`  
`bool operator== (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept`
- `template<typename _Tp, typename _Dp >`  
`bool operator!= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x) noexcept`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename _Dp, typename _Up, typename _Ep >`  
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)`
- `template<typename _Tp, typename _Dp >`  
`bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)`
- `template<typename _Tp, typename _Dp >`  
`bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq< _Tp >::__single_object make_unique (_Args &&... __args)`
- `template<typename _Tp >`  
`_MakeUniq< _Tp >::__array make_unique (size_t __num)`
- `template<typename _Tp, typename... _Args>`  
`_MakeUniq< _Tp >::__invalid_type make_unique (_Args &&...)=delete`

#### 4.1032.1 Detailed Description

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr< _Tp, _Dp >
```

20.7.1.2 unique\_ptr for single objects.

Definition at line 242 of file unique\_ptr.h.

## 4.1032.2 Constructor &amp; Destructor Documentation

## 4.1032.2.1 unique\_ptr() [1/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
constexpr std::unique_ptr< _Tp, _Dp >::unique_ptr () [inline], [noexcept]
```

Default constructor, creates a unique\_ptr that owns nothing.


Definition at line 269 of file unique\_ptr.h.

## 4.1032.2.2 unique\_ptr() [2/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p) [inline], [explicit], [noexcept]
```

Takes ownership of a pointer.

## Parameters

|                                                                                            |                                        |
|--------------------------------------------------------------------------------------------|----------------------------------------|
| <br>__p | A pointer to an object of element_type |
|--------------------------------------------------------------------------------------------|----------------------------------------|

The deleter will be value-initialized.



Definition at line 281 of file unique\_ptr.h.

## 4.1032.2.3 unique\_ptr() [3/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p,
 const deleter_type & __d) [inline], [noexcept]
```

Takes ownership of a pointer.

## Parameters

|                                                                                            |                                        |
|--------------------------------------------------------------------------------------------|----------------------------------------|
| <br>__p | A pointer to an object of element_type |
| <br>__d | A reference to a deleter.              |
| Generated by Doxygen                                                                       |                                        |

The deleter will be initialized with `__d`

Definition at line 294 of file `unique_ptr.h`.

#### 4.1032.2.4 `unique_ptr()` [4/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 pointer __p,
 __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [noexcept]
```

Takes ownership of a pointer.

##### Parameters

|                  |                                                     |
|------------------|-----------------------------------------------------|
| <code>__p</code> | A pointer to an object of <code>element_type</code> |
| <code>__d</code> | An rvalue reference to a (non-reference) deleter.   |

The deleter will be initialized with `std::move(__d)`

Definition at line 306 of file `unique_ptr.h`.

#### 4.1032.2.5 `unique_ptr()` [5/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
constexpr std::unique_ptr< _Tp, _Dp >::unique_ptr (
 nullptr_t) [inline], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

Definition at line 320 of file `unique_ptr.h`.

#### 4.1032.2.6 `unique_ptr()` [6/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 unique_ptr< _Tp, _Dp > &&) [default]
```

Move constructor.



## 4.1032.2.7 unique\_ptr() [7/7]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>,
typename conditional<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>><_
::type>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
 unique_ptr< _Up, _Ep > && __u) [inline], [noexcept]
```

Converting constructor from another type.

Requires that the pointer owned by \_\_u is convertible to the type of pointer owned by this object, \_\_u does not own an array, and \_\_u has a compatible deleter type.

Definition at line 340 of file unique\_ptr.h.

## 4.1032.2.8 ~unique\_ptr()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::~~unique_ptr () [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 355 of file unique\_ptr.h.

## 4.1032.3 Member Function Documentation

## 4.1032.3.1 get()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::get (
 void) const [inline], [noexcept]
```

Return the stored pointer.

Definition at line 421 of file unique\_ptr.h.

Referenced by std::unique\_ptr< \_Result< \_Res > >::operator bool(), std::unique\_ptr< \_Tp[], \_Dp >::operator bool(), std::unique\_ptr< \_Result< \_Res > >::operator!(), std::unique\_ptr< \_Result< \_Res > >::operator<(), std::unique\_ptr< \_Result< \_Res > >::operator==(), and std::unique\_ptr< \_Result< \_Res > >::operator>().

**4.1032.3.2** `get_deleter()` [1/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type& std::unique_ptr< _Tp, _Dp >::get_deleter () [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 426 of file `unique_ptr.h`.

Referenced by `std::unique_ptr< _Result< _Res > >::operator=()`, `std::unique_ptr< _Tp[], _Dp >::operator=()`, `std::unique_ptr< _Result< _Res > >::~~unique_ptr()`, and `std::unique_ptr< _Tp[], _Dp >::~~unique_ptr()`.

**4.1032.3.3** `get_deleter()` [2/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type& std::unique_ptr< _Tp, _Dp >::get_deleter () const [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 431 of file `unique_ptr.h`.

**4.1032.3.4** `operator bool()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::operator bool () const [inline], [explicit], [noexcept]
```

Return `true` if the stored pointer is not null.

Definition at line 435 of file `unique_ptr.h`.

**4.1032.3.5** `operator*()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference<element_type>::type std::unique_ptr< _Tp, _Dp >::operator* () const [inline]
```

Dereference the stored pointer.

Definition at line 405 of file `unique_ptr.h`.

**4.1032.3.6 operator->()**

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::operator-> () const [inline], [noexcept]
```

Return the stored pointer.

Definition at line 413 of file unique\_ptr.h.

**4.1032.3.7 operator=()** [1/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr& std::unique_ptr<_Tp, _Dp>::operator= (
 unique_ptr<_Tp, _Dp> &&) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

**4.1032.3.8 operator=()** [2/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep >
enable_if< __and< __safe_conversion_up<_Up, _Ep>, is_assignable<deleter_type&, _Ep&&> >&
::value, unique_ptr&::type std::unique_ptr<_Tp, _Dp>::operator= (
 unique_ptr<_Up, _Ep> && __u) [inline], [noexcept]
```

Assignment from another type.

**Parameters**

|                         |                                                                                                |
|-------------------------|------------------------------------------------------------------------------------------------|
| <b><code>__u</code></b> | The object to transfer ownership from, which owns a convertible pointer to a non-array object. |
|-------------------------|------------------------------------------------------------------------------------------------|

Invokes the deleter if this object owns a pointer.

Definition at line 386 of file unique\_ptr.h.

**4.1032.3.9 operator=()** [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr& std::unique_ptr<_Tp, _Dp>::operator= (
 nullptr_t) [inline], [noexcept]
```

Reset the unique\_ptr to empty, invoking the deleter if necessary.

Definition at line 395 of file unique\_ptr.h.

#### 4.1032.3.10 `release()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr<_Tp, _Dp>::release () [inline], [noexcept]
```

Release ownership of any stored pointer.

Definition at line 442 of file `unique_ptr.h`.

#### 4.1032.3.11 `reset()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::reset (
 pointer __p = pointer()) [inline], [noexcept]
```

Replace the stored pointer.

##### Parameters

|                  |                           |
|------------------|---------------------------|
| <code>__p</code> | The new pointer to store. |
|------------------|---------------------------|

The deleter will be invoked if a pointer is already owned.

Definition at line 452 of file `unique_ptr.h`.

Referenced by `std::unique_ptr<_Result<_Res>>::operator=()`, and `std::unique_ptr<_Tp[], _Dp>::operator=()`.

#### 4.1032.3.12 `swap()`

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr<_Tp, _Dp>::swap (
 unique_ptr<_Tp, _Dp> & __u) [inline], [noexcept]
```

Exchange the pointer and deleter with another object.

Definition at line 461 of file `unique_ptr.h`.

The documentation for this class was generated from the following files:

- [unique\\_ptr.h](#)
- [auto\\_ptr.h](#)

## 4.1033 std::unique\_ptr&lt; \_Tp[], \_Dp &gt; Class Template Reference

## Public Types

- template<typename \_Up >  
using **\_\_safe\_conversion\_raw** = \_\_and\_< \_\_or\_< \_\_or\_< is\_same< \_Up, pointer >, is\_same< \_Up, nullptr > >, \_\_and\_< is\_pointer< \_Up >, is\_same< pointer, element\_type \* >, is\_convertible< typename remove\_pointer< \_Up >::type(\*)[], element\_type(\*)[] > > > >
- template<typename \_Up, typename \_Ep, typename \_UPtr = unique\_ptr< \_Up, \_Ep >, typename \_UP\_pointer = typename \_UPtr::pointer, typename \_UP\_element\_type = typename \_UPtr::element\_type >  
using **\_\_safe\_conversion\_up** = \_\_and\_< is\_array< \_Up >, is\_same< pointer, element\_type \* >, is\_same< \_UP\_pointer, \_UP\_element\_type \* >, is\_convertible< \_UP\_element\_type(\*)[], element\_type(\*)[] > >
- using **deleter\_type** = \_Dp
- using **element\_type** = \_Tp
- using **pointer** = typename \_\_uniq\_ptr\_impl< \_Tp, \_Dp >::pointer

## Public Member Functions

- template<typename \_Del = \_Dp, typename = \_DeleterConstraint< \_Del >>  
constexpr **unique\_ptr** () noexcept
- template<typename \_Up, typename \_Vp = \_Dp, typename = \_DeleterConstraint< \_Vp >, typename = typename enable\_if< \_\_safe\_conversion\_raw< \_Up >::value, bool >::type >  
**unique\_ptr** ( \_Up \_\_p) noexcept
- template<typename \_Up, typename \_Del = deleter\_type, typename = \_Require< \_\_safe\_conversion\_raw< \_Up >, is\_copy\_constructible< \_Del > >>  
**unique\_ptr** ( \_Up \_\_p, const deleter\_type & \_\_d) noexcept
- template<typename \_Up, typename \_Del = deleter\_type, typename = \_Require< \_\_safe\_conversion\_raw< \_Up >, is\_move\_constructible< \_Del > >>  
**unique\_ptr** ( \_Up \_\_p, \_\_enable\_if\_t< !is\_lvalue\_reference< \_Del >::value, \_Del && > \_\_d) noexcept
- template<typename \_Up, typename \_Del = deleter\_type, typename \_DelUnref = typename remove\_reference< \_Del >::type, typename = \_Require< \_\_safe\_conversion\_raw< \_Up > >>  
**unique\_ptr** ( \_Up, \_\_enable\_if\_t< is\_lvalue\_reference< \_Del >::value, \_DelUnref && >) = delete
- **unique\_ptr** ( **unique\_ptr** &&) = default
- template<typename \_Del = \_Dp, typename = \_DeleterConstraint< \_Del >>  
constexpr **unique\_ptr** ( nullptr\_t) noexcept
- template<typename \_Up, typename \_Ep, typename = \_Require< \_\_safe\_conversion\_up< \_Up, \_Ep >, typename conditional< is\_lvalue\_reference< \_Dp >::value, is\_same< \_Ep, \_Dp >, is\_convertible< \_Ep, \_Dp > >::type >>  
**unique\_ptr** ( **unique\_ptr**< \_Up, \_Ep > && \_\_u) noexcept
- **unique\_ptr** ( const **unique\_ptr** &) = delete
- **~unique\_ptr** ()
- pointer **get** () const noexcept
- deleter\_type & **get\_deleter** () noexcept
- const deleter\_type & **get\_deleter** () const noexcept
- **operator bool** () const noexcept
- **unique\_ptr** & **operator=** ( **unique\_ptr** &&) = default
- template<typename \_Up, typename \_Ep >  
**enable\_if**< \_\_and\_< \_\_safe\_conversion\_up< \_Up, \_Ep >, is\_assignable< deleter\_type &, \_Ep && > >::value, **unique\_ptr** & >::type **operator=** ( **unique\_ptr**< \_Up, \_Ep > && \_\_u) noexcept
- **unique\_ptr** & **operator=** ( nullptr\_t) noexcept
- **unique\_ptr** & **operator=** ( const **unique\_ptr** &) = delete
- **std::add\_lvalue\_reference**< element\_type >::type **operator[]** ( size\_t \_\_i) const

- pointer `release` () noexcept
- `template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer, element_type*>, is_convertible< typename remove_pointer<_Up>::type(*)[], element_type(*)[] > > >>>`  
`void reset (_Up __p) noexcept`
- `void reset (nullptr_t=nullptr) noexcept`
- `void swap (unique_ptr &__u) noexcept`

#### 4.1033.1 Detailed Description

```
template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[], _Dp >
```

##### 20.7.1.3 unique\_ptr for array objects with a runtime length

Definition at line 477 of file `unique_ptr.h`.

#### 4.1033.2 Constructor & Destructor Documentation

##### 4.1033.2.1 unique\_ptr() [1/6]

```
template<typename _Tp, typename _Dp >
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr () [inline], [noexcept]
```

Default constructor, creates a `unique_ptr` that owns nothing.

Definition at line 530 of file `unique_ptr.h`.

##### 4.1033.2.2 unique\_ptr() [2/6]

```
template<typename _Tp, typename _Dp >
template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p) [inline], [explicit], [noexcept]
```

Takes ownership of a pointer.

#### Parameters

|                  |                                                                                             |
|------------------|---------------------------------------------------------------------------------------------|
| <code>__p</code> | A pointer to an array of a type safely convertible to an array of <code>element_type</code> |
|------------------|---------------------------------------------------------------------------------------------|

The deleter will be value-initialized.

Definition at line 547 of file unique\_ptr.h.

#### 4.1033.2.3 unique\_ptr() [3/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename _Del = deleter_type, typename = _Require<__safe_conversion_↵
raw<_Up>, is_copy_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p,
 const deleter_type & __d) [inline], [noexcept]
```

Takes ownership of a pointer.

##### Parameters

|                    |                                                                                |
|--------------------|--------------------------------------------------------------------------------|
| $\leftarrow$<br>_p | A pointer to an array of a type safely convertible to an array of element_type |
| $\leftarrow$<br>_d | A reference to a deleter.                                                      |

The deleter will be initialized with \_\_d

Definition at line 562 of file unique\_ptr.h.

#### 4.1033.2.4 unique\_ptr() [4/6]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename _Del = deleter_type, typename = _Require<__safe_conversion_↵
raw<_Up>, is_move_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 _Up __p,
 __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) [inline], [noexcept]
```

Takes ownership of a pointer.

##### Parameters

|                    |                                                                                |
|--------------------|--------------------------------------------------------------------------------|
| $\leftarrow$<br>_p | A pointer to an array of a type safely convertible to an array of element_type |
| $\leftarrow$<br>_d | A reference to a deleter.                                                      |

The deleter will be initialized with std::move(\_\_d)

Definition at line 576 of file `unique_ptr.h`.

#### 4.1033.2.5 `unique_ptr()` [5/6]

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 unique_ptr< _Tp[], _Dp > &&) [default]
```

Move constructor.

#### 4.1033.2.6 `unique_ptr()` [6/6]

```
template<typename _Tp , typename _Dp >
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
constexpr std::unique_ptr< _Tp[], _Dp >::unique_ptr (
 nullptr_t) [inline], [noexcept]
```

Creates a `unique_ptr` that owns nothing.

Definition at line 594 of file `unique_ptr.h`.

#### 4.1033.2.7 `~unique_ptr()`

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::~~unique_ptr () [inline]
```

Destructor, invokes the deleter if the stored pointer is not null.

Definition at line 608 of file `unique_ptr.h`.

References `std::unique_ptr< _Tp, _Dp >::get_deleter()`.

### 4.1033.3 Member Function Documentation

#### 4.1033.3.1 `get()`

```
template<typename _Tp , typename _Dp >
pointer std::unique_ptr< _Tp[], _Dp >::get (
 void) const [inline], [noexcept]
```

Return the stored pointer.

Definition at line 665 of file `unique_ptr.h`.



## 4.1033.3.2 get\_deleter() [1/2]

```
template<typename _Tp , typename _Dp >
deleter_type& std::unique_ptr< _Tp[], _Dp >::get_deleter () [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 670 of file unique\_ptr.h.

## 4.1033.3.3 get\_deleter() [2/2]

```
template<typename _Tp , typename _Dp >
const deleter_type& std::unique_ptr< _Tp[], _Dp >::get_deleter () const [inline], [noexcept]
```

Return a reference to the stored deleter.

Definition at line 675 of file unique\_ptr.h.

## 4.1033.3.4 operator bool()

```
template<typename _Tp , typename _Dp >
std::unique_ptr< _Tp[], _Dp >::operator bool () const [inline], [explicit], [noexcept]
```

Return `true` if the stored pointer is not null.

Definition at line 679 of file unique\_ptr.h.

References `std::unique_ptr< _Tp, _Dp >::get()`.

## 4.1033.3.5 operator=() [1/3]

```
template<typename _Tp , typename _Dp >
unique_ptr& std::unique_ptr< _Tp[], _Dp >::operator= (
 unique_ptr< _Tp[], _Dp > &&) [default]
```

Move assignment operator.

Invokes the deleter if this object owns a pointer.

## 4.1033.3.6 operator=() [2/3]

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename _Ep >
enable_if<__and<__safe_conversion_up<_Up, _Ep>, is_assignable<deleter_type&, _Ep&&> >::value,
unique_ptr&>::type std::unique_ptr< _Tp[], _Dp >::operator= (
 unique_ptr< _Up, _Ep > && __u) [inline], [noexcept]
```

Assignment from another type.

## Parameters

|                                   |                                                                                             |
|-----------------------------------|---------------------------------------------------------------------------------------------|
| <code>↵</code><br><code>_U</code> | The object to transfer ownership from, which owns a convertible pointer to an array object. |
|-----------------------------------|---------------------------------------------------------------------------------------------|

Invokes the deleter if this object owns a pointer.

Definition at line 638 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::get_deleter()`, and `std::unique_ptr<_Tp, _Dp>::reset()`.

4.1033.3.7 `operator=()` [3/3]

```
template<typename _Tp , typename _Dp >
unique_ptr& std::unique_ptr< _Tp[], _Dp >::operator= (
 nullptr_t) [inline], [noexcept]
```

Reset the `unique_ptr` to empty, invoking the deleter if necessary.

Definition at line 647 of file `unique_ptr.h`.

References `std::unique_ptr<_Tp, _Dp>::reset()`.

4.1033.3.8 `operator[]()`

```
template<typename _Tp , typename _Dp >
std::add_lvalue_reference<element_type>::type std::unique_ptr< _Tp[], _Dp >::operator[] (
 size_t __i) const [inline]
```

Access an element of owned array.

Definition at line 657 of file `unique_ptr.h`.

4.1033.3.9 `release()`

```
template<typename _Tp , typename _Dp >
pointer std::unique_ptr< _Tp[], _Dp >::release () [inline], [noexcept]
```

Release ownership of any stored pointer.


Definition at line 686 of file `unique_ptr.h`.

4.1033.3.10 `reset()`

```
template<typename _Tp , typename _Dp >
template<typename _Up , typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element↵
_type(*)[] > > > >>
void std::unique_ptr< _Tp[], _Dp >::reset (
 _Up __p) [inline], [noexcept]
```

Replace the stored pointer.

## Parameters

|                                                                                                   |                           |
|---------------------------------------------------------------------------------------------------|---------------------------|
| <a href="#"></a> | The new pointer to store. |
| <a href="#">_p</a>                                                                                |                           |

The deleter will be invoked if a pointer is already owned.

Definition at line 708 of file unique\_ptr.h.

References [std::move\(\)](#).

## 4.1033.3.11 swap()

```
template<typename _Tp , typename _Dp >
void std::unique_ptr< _Tp[], _Dp >::swap (
 unique_ptr< _Tp[], _Dp > & __u) [inline], [noexcept]
```

Exchange the pointer and deleter with another object.

Definition at line 716 of file unique\_ptr.h.

The documentation for this class was generated from the following file:

- [unique\\_ptr.h](#)

## 4.1034 std::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
  - typedef \_Hashtable::value\_type [value\\_type](#)
  - typedef \_Hashtable::mapped\_type [mapped\\_type](#)
  - typedef \_Hashtable::hasher [hasher](#)
  - typedef \_Hashtable::key\_equal [key\\_equal](#)
  - typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- 
- typedef \_Hashtable::pointer [pointer](#)
  - typedef \_Hashtable::const\_pointer [const\\_pointer](#)
  - typedef \_Hashtable::reference [reference](#)
  - typedef \_Hashtable::const\_reference [const\\_reference](#)
  - typedef \_Hashtable::iterator [iterator](#)
  - typedef \_Hashtable::const\_iterator [const\\_iterator](#)
  - typedef \_Hashtable::local\_iterator [local\\_iterator](#)
  - typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
  - typedef \_Hashtable::size\_type [size\\_type](#)
  - typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- `unordered_map` ()=default
- `unordered_map` (`size_type` \_\_n, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_map` (const `unordered_map` &)=default
- `unordered_map` (`unordered_map` &&)=default
- `unordered_map` (const `allocator_type` &\_\_a)
- `unordered_map` (const `unordered_map` &\_\_umap, const `allocator_type` &\_\_a)
- `unordered_map` (`unordered_map` &&\_\_umap, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n=0, const `hasher` &\_\_hf=`hasher`(), const `key_equal` &\_\_eq=`key_equal`(), const `allocator_type` &\_\_a=`allocator_type`())
- `unordered_map` (`size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_map` (`size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `template<typename _InputIterator >`  
`unordered_map` (\_InputIterator \_\_first, \_InputIterator \_\_last, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `allocator_type` &\_\_a)
- `unordered_map` (`initializer_list`< `value_type` > \_\_l, `size_type` \_\_n, const `hasher` &\_\_hf, const `allocator_type` &\_\_a)
- `iterator begin` () noexcept
- `local_iterator begin` (`size_type` \_\_n)
- `size_type bucket` (const `key_type` &\_\_key) const
- `size_type bucket_count` () const noexcept
- `size_type bucket_size` (`size_type` \_\_n) const
- `void clear` () noexcept
- `size_type count` (const `key_type` &\_\_x) const
- `template<typename... _Args>`  
`std::pair`< `iterator`, bool > `emplace` (\_Args &&... \_\_args)
- `template<typename... _Args>`  
`iterator emplace_hint` (const `iterator` \_\_pos, \_Args &&... \_\_args)
- `bool empty` () const noexcept
- `iterator end` () noexcept
- `local_iterator end` (`size_type` \_\_n)
- `size_type erase` (const `key_type` &\_\_x)
- `iterator erase` (const `iterator` \_\_first, const `iterator` \_\_last)
- `allocator_type get_allocator` () const noexcept
- `hasher hash_function` () const
- `template<typename _InputIterator >`  
`void insert` (\_InputIterator \_\_first, \_InputIterator \_\_last)
- `void insert` (`initializer_list`< `value_type` > \_\_l)
- `key_equal key_eq` () const
- `float load_factor` () const noexcept
- `size_type max_bucket_count` () const noexcept
- `float max_load_factor` () const noexcept
- `void max_load_factor` (float \_\_z)



- `std::pair< iterator, iterator > equal_range` (const `key_type` &\_\_x)
- `std::pair< const_iterator, const_iterator > equal_range` (const `key_type` &\_\_x) const

- `mapped_type & operator[]` (const `key_type` &\_\_k)
- `mapped_type & operator[]` (`key_type` &&\_\_k)

- `mapped_type & at` (const `key_type` &\_\_k)
- `const mapped_type & at` (const `key_type` &\_\_k) const

- `const_local_iterator begin` (`size_type` \_\_n) const
- `const_local_iterator cbegin` (`size_type` \_\_n) const

- `const_local_iterator end` (`size_type` \_\_n) const
- `const_local_iterator cend` (`size_type` \_\_n) const

#### Friends

- `template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1 >`  
`bool operator==` (const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &, const `unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 >` &)

#### 4.1034.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc =
allocator<std::pair<const _Key, _Tp>>>>
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

## Template Parameters

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .                       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .                 |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a [container](#), and [unordered associative container](#)

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__umap_hashtable`.

Definition at line 102 of file `unordered_map.h`.

## 4.1034.2 Member Typedef Documentation

## 4.1034.2.1 allocator\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

Definition at line 116 of file `unordered_map.h`.

## 4.1034.2.2 const\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

Definition at line 126 of file `unordered_map.h`.

#### 4.1034.2.3 const\_local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

Definition at line 128 of file unordered\_map.h.

#### 4.1034.2.4 const\_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 122 of file unordered\_map.h.

#### 4.1034.2.5 const\_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 124 of file unordered\_map.h.

#### 4.1034.2.6 difference\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::difference_type
```

Iterator-related typedefs.

Definition at line 130 of file unordered\_map.h.



#### 4.1034.2.7 hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 114 of file unordered\_map.h.

#### 4.1034.2.8 iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 125 of file unordered\_map.h.

#### 4.1034.2.9 key\_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 115 of file unordered\_map.h.

#### 4.1034.2.10 key\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 111 of file unordered\_map.h.

#### 4.1034.2.11 local\_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::local_iterator
```

Iterator-related typedefs.

Definition at line 127 of file unordered\_map.h.

#### 4.1034.2.12 mapped\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::mapped_type
```

Public typedefs.

Definition at line 113 of file unordered\_map.h.

#### 4.1034.2.13 pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 121 of file unordered\_map.h.

#### 4.1034.2.14 reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 123 of file unordered\_map.h.

## 4.1034.2.15 size\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 129 of file unordered\_map.h.

## 4.1034.2.16 value\_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::value_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 112 of file unordered\_map.h.

## 4.1034.3 Constructor &amp; Destructor Documentation

## 4.1034.3.1 unordered\_map() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map () [default]
```

Default constructor.

## 4.1034.3.2 unordered\_map() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Definition at line 151 of file `unordered_map.h`.

4.1034.3.3 `unordered_map()` [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_map` from a range.

## Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_map` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

Definition at line 172 of file `unordered_map.h`.

4.1034.3.4 `unordered_map()` [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

## 4.1034.3.5 unordered\_map() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

## 4.1034.3.6 unordered\_map() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_map with no elements.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Definition at line 191 of file unordered\_map.h.

## 4.1034.3.7 unordered\_map() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_map from an initializer\_list.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an `unordered_map` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `__l.size()`).

Definition at line 226 of file `unordered_map.h`.

#### 4.1034.4 Member Function Documentation

##### 4.1034.4.1 `at()` [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at (
 const key_type & __k) [inline]
```

Access to `unordered_map` data.

##### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

##### Returns

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

##### Exceptions

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 1000 of file `unordered_map.h`.

##### 4.1034.4.2 `at()` [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::at (
 const key_type & __k) const [inline]
```

Access to `unordered_map` data.

##### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__k</code> | The key for which data should be retrieved. |
|------------------|---------------------------------------------|

**Returns**

A reference to the data whose key is equal to `__k`, if such a data is present in the `unordered_map`.

**Exceptions**

|                                |                             |
|--------------------------------|-----------------------------|
| <code>std::out_of_range</code> | If no such data is present. |
|--------------------------------|-----------------------------|

Definition at line 1004 of file `unordered_map.h`.

**4.1034.4.3 begin()** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the `unordered_map`.

Definition at line 324 of file `unordered_map.h`.

**4.1034.4.4 begin()** [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_map`.

Definition at line 333 of file `unordered_map.h`.

**4.1034.4.5 begin()** [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read/write local iterator.

Definition at line 1045 of file unordered\_map.h.

**4.1034.4.6 begin()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

Definition at line 1056 of file unordered\_map.h.

**4.1034.4.7 bucket\_count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_map.

Definition at line 1012 of file unordered\_map.h.

**4.1034.4.8 cbegin()** [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_map.

Definition at line 337 of file unordered\_map.h.



## 4.1034.4.9 cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

## Returns

A read-only local iterator.

Definition at line 1060 of file unordered\_map.h.

## 4.1034.4.10 cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_map.

Definition at line 359 of file unordered\_map.h.

## 4.1034.4.11 cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

Definition at line 1086 of file unordered\_map.h.

**4.1034.4.12 clear()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 841 of file unordered\_map.h.

**4.1034.4.13 count()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |               |
|------------------|---------------|
| <code>__x</code> | Key to count. |
|------------------|---------------|

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multimap; for unordered\_map the result will either be 0 (not present) or 1 (present).

Definition at line 937 of file unordered\_map.h.

4.1034.4.14 `emplace()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
std::pair<iterator, bool> std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

## Parameters

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 387 of file `unordered_map.h`.

4.1034.4.15 `emplace_hint()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_map`.

## Parameters

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args` (may or may not be that `std::pair`).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 418 of file `unordered_map.h`.

#### 4.1034.4.16 `empty()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::empty () const [inline], [noexcept]
```

Returns true if the `unordered_map` is empty.

Definition at line 304 of file `unordered_map.h`.

#### 4.1034.4.17 `end()` [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_map`.

Definition at line 346 of file `unordered_map.h`.

#### 4.1034.4.18 `end()` [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_map`.

Definition at line 355 of file `unordered_map.h`.

#### 4.1034.4.19 `end()` [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::end (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

## Returns

A read/write local iterator.

Definition at line 1071 of file unordered\_map.h.

## 4.1034.4.20 end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

## Returns

A read-only local iterator.

Definition at line 1082 of file unordered\_map.h.

## 4.1034.4.21 equal\_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range
(
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

Definition at line 961 of file `unordered_map.h`.

**4.1034.4.22 `equal_range()`** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<const_iterator, const_iterator> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for `unordered_multimap`.

Definition at line 965 of file `unordered_map.h`.

**4.1034.4.23 `erase()`** [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an `unordered_map`.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 791 of file `unordered_map.h`.

4.1034.4.24 `erase()` [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an `unordered_map`.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

## Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_map`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 796 of file `unordered_map.h`.

4.1034.4.25 `erase()` [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                 |                              |
|-----------------|------------------------------|
| <code>_↵</code> | Key of element to be erased. |
| <code>_X</code> |                              |

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_map`. For an `unordered_map` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 813 of file `unordered_map.h`.

**4.1034.4.26 erase()** [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_map`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_map`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 831 of file `unordered_map.h`.



## 4.1034.4.27 find() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an unordered\_map.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 919 of file `unordered_map.h`.

**4.1034.4.28 find()** [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an `unordered_map`.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_X</code> |                    |

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 923 of file `unordered_map.h`.

**4.1034.4.29 get\_allocator()**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the `unordered_map`.

Definition at line 297 of file `unordered_map.h`.

## 4.1034.4.30 hash\_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the unordered\_map was constructed.

Definition at line 895 of file unordered\_map.h.

## 4.1034.4.31 insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| <code>__x</code> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|------------------|----------------------------------------------------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered\_map. An unordered\_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered\_map.

Insertion requires amortized constant time.

Definition at line 579 of file unordered\_map.h.

## 4.1034.4.32 insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, bool> std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

**Parameters**

|                                    |                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>_↵</code><br><code>_X</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 585 of file `unordered_map.h`.

**4.1034.4.33 insert()** [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↵
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, pair<iterator, bool> > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                                    |                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------|
| <code>_↵</code><br><code>_X</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------------------|-----------------------------------------------------------------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the `unordered_map`. An `unordered_map` relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the `unordered_map`.

Insertion requires amortized constant time.

Definition at line 591 of file `unordered_map.h`.

## 4.1034.4.34 insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 618 of file unordered\_map.h.

## 4.1034.4.35 insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Attempts to insert a std::pair into the unordered\_map.

## Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 624 of file `unordered_map.h`.

**4.1034.4.36 insert()** [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_map< _Key,
_Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 _Pair && __x) [inline]
```

Attempts to insert a `std::pair` into the `unordered_map`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 629 of file `unordered_map.h`.

## 4.1034.4.37 insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 644 of file unordered\_map.h.

## 4.1034.4.38 insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the unordered\_map.

## Parameters

|                                                                                        |                                                                 |
|----------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| <code>↵</code><br><code>↵</code><br><code>↵</code><br><code>↵</code><br><code>/</code> | A std::initializer_list<value_type> of elements to be inserted. |
|----------------------------------------------------------------------------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 655 of file unordered\_map.h.

## 4.1034.4.39 key\_eq()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the `unordered_map` was constructed.

Definition at line 901 of file `unordered_map.h`.

#### 4.1034.4.40 `load_factor()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1094 of file `unordered_map.h`.

#### 4.1034.4.41 `max_bucket_count()`

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the `unordered_map`.

Definition at line 1017 of file `unordered_map.h`.

#### 4.1034.4.42 `max_load_factor()` [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the `unordered_map` tries to keep the load factor less than or equal to.

Definition at line 1100 of file `unordered_map.h`.

#### 4.1034.4.43 `max_load_factor()` [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the `unordered_map` maximum load factor.



## Parameters

|                 |                              |
|-----------------|------------------------------|
| <code>_Z</code> | The new maximum load factor. |
|-----------------|------------------------------|

Definition at line 1108 of file unordered\_map.h.

## 4.1034.4.44 max\_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_map.

Definition at line 314 of file unordered\_map.h.

## 4.1034.4.45 operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

## 4.1034.4.46 operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

## 4.1034.4.47 operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map& std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > & __l) [inline]
```

Unordered\_map list assignment operator.

## Parameters

|                |                      |
|----------------|----------------------|
| <code>↵</code> | An initializer_list. |
| <code>↵</code> |                      |
| <code>↵</code> |                      |
| <code>↵</code> |                      |
| <code>/</code> |                      |

This function fills an unordered\_map with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the unordered\_map and that the resulting unordered\_map's size is the same as the number of elements assigned.

Definition at line 289 of file unordered\_map.h.

4.1034.4.48 `operator[]()` [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[] (
 const key_type & __k) [inline]
```

Subscript ( `[]` ) access to unordered\_map data.

## Parameters

|                                   |                                             |
|-----------------------------------|---------------------------------------------|
| <code>↵</code><br><code>_k</code> | The key for which data should be retrieved. |
|-----------------------------------|---------------------------------------------|

## Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( `[]` ) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 983 of file unordered\_map.h.

4.1034.4.49 `operator[]()` [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type& std::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>::operator[] (
 key_type && __k) [inline]
```

Subscript ( `[]` ) access to unordered\_map data.

## Parameters

|                       |                                             |
|-----------------------|---------------------------------------------|
| <code>↵<br/>_k</code> | The key for which data should be retrieved. |
|-----------------------|---------------------------------------------|

## Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ( [] )operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.

Lookup requires constant time.

Definition at line 987 of file unordered\_map.h.

## 4.1034.4.50 rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_map.

## Parameters

|                       |                            |
|-----------------------|----------------------------|
| <code>↵<br/>_n</code> | The new number of buckets. |
|-----------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_map maximum load factor.

Definition at line 1119 of file unordered\_map.h.

## 4.1034.4.51 reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_map for a specified number of elements.

**Parameters**

|                        |                              |
|------------------------|------------------------------|
| <code>_↔<br/>_n</code> | Number of elements required. |
|------------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1130 of file `unordered_map.h`.

**4.1034.4.52 `size()`**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the `unordered_map`.

Definition at line 309 of file `unordered_map.h`.

**4.1034.4.53 `swap()`**

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↔
_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
 unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another `unordered_map`.

**Parameters**

|                        |                                                                        |
|------------------------|------------------------------------------------------------------------|
| <code>_↔<br/>_X</code> | An <code>unordered_map</code> of the same element and allocator types. |
|------------------------|------------------------------------------------------------------------|

This exchanges the elements between two `unordered_map` in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

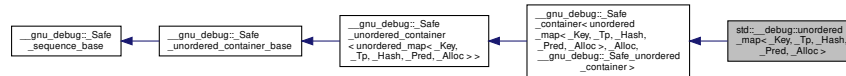
Definition at line 855 of file `unordered_map.h`.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

## 4.1035 std::\_\_debug::unordered\_map&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >:



## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::Safe\_iterator< \_Base\_const\_iterator, unordered\_map > **const\_iterator**
- typedef \_\_gnu\_debug::Safe\_local\_iterator< \_Base\_const\_local\_iterator, unordered\_map > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef \_\_gnu\_debug::Safe\_iterator< \_Base\_iterator, unordered\_map > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_\_gnu\_debug::Safe\_local\_iterator< \_Base\_local\_iterator, unordered\_map > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (const unordered\_map &)=default
- **unordered\_map** (const \_Base &\_\_x)
- **unordered\_map** (unordered\_map &&)=default
- **unordered\_map** (const allocator\_type &\_\_a)
- **unordered\_map** (const unordered\_map &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** (unordered\_map &&\_\_umap, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_map** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_map** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_map** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)

- `_Base & _M_base ()` noexcept
- `const _Base & _M_base ()` const noexcept
- `void _M_swap (_Safe_container &__x)` noexcept
- `iterator begin ()` noexcept
- `const_iterator begin ()` const noexcept
- `local_iterator begin (size_type __b)`
- `const_local_iterator begin (size_type __b)` const
- `size_type bucket_size (size_type __b)` const
- `const_iterator cbegin ()` const noexcept
- `const_local_iterator cbegin (size_type __b)` const
- `const_iterator cend ()` const noexcept
- `const_local_iterator cend (size_type __b)` const
- `void clear ()` noexcept
- `template<typename... _Args>`  
`std::pair< iterator, bool > emplace (_Args &&... __args)`
- `template<typename... _Args>`  
`iterator emplace_hint (const_iterator __hint, _Args &&... __args)`
- `iterator end ()` noexcept
- `const_iterator end ()` const noexcept
- `local_iterator end (size_type __b)`
- `const_local_iterator end (size_type __b)` const
- `std::pair< iterator, iterator > equal_range (const key_type &__key)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__key)` const
- `size_type erase (const key_type &__key)`
- `iterator erase (const_iterator __it)`
- `iterator erase (iterator __it)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `iterator find (const key_type &__key)`
- `const_iterator find (const key_type &__key)` const
- `std::pair< iterator, bool > insert (const value_type &__obj)`
- `std::pair< iterator, bool > insert (value_type &&__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`std::pair< iterator, bool > insert (_Pair &&__obj)`
- `iterator insert (const_iterator __hint, const value_type &__obj)`
- `iterator insert (const_iterator __hint, value_type &&__x)`
- `template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>`  
`iterator insert (const_iterator __hint, _Pair &&__obj)`
- `void insert (std::initializer_list< value_type > __l)`
- `template<typename _InputIterator >`  
`void insert (_InputIterator __first, _InputIterator __last)`
- `float max_load_factor ()` const noexcept
- `void max_load_factor (float __f)`
- `unordered_map & operator= (const unordered_map &)=default`
- `unordered_map & operator= (unordered_map &&)=default`
- `unordered_map & operator= (initializer_list< value_type > __l)`
- `void swap (unordered_map &__x)` noexcept(noexcept(declval< \_Base & >().swap(\_\_x)))

## Public Attributes

- `_Safe_iterator_base` \* [\\_M\\_const\\_iterators](#)
- `_Safe_iterator_base` \* [\\_M\\_const\\_local\\_iterators](#)
- `_Safe_iterator_base` \* [\\_M\\_iterators](#)
- `_Safe_iterator_base` \* [\\_M\\_local\\_iterators](#)
- unsigned int [\\_M\\_version](#)

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- `__gnu_cxx::__mutex` & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_local\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_locals](#) ()
- void [\\_M\\_revalidate\\_singular](#) ()
- `_Safe_container` & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) (\_Safe\_unordered\_container\_base &\_\_x) noexcept
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x) noexcept

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
class `::__gnu_debug::Safe_iterator`
- `template<typename _ItT, typename _SeqT>`  
class `::__gnu_debug::Safe_local_iterator`

## 4.1035.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class `std::unordered_map` with safety/checking/debug instrumentation.

Definition at line 41 of file `debug/unordered_map`.

## 4.1035.2 Member Function Documentation

#### 4.1035.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### 4.1035.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators *i* attached to this sequence, *i*->\_M\_version == \_M\_version.

#### 4.1035.2.3 `_M_get_mutex()`

```
__gnu_cxx::mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1035.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.1035.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
>::_M_invalidate_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all iterators *x* that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.



## 4.1035.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
>::_M_invalidate_local_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

## 4.1035.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.1035.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.1035.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.1035.3 Member Data Documentation

## 4.1035.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1035.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 4.1035.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1035.3.4 `_M_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

#### 4.1035.3.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 4.1036 std::unordered\_multimap&lt; \_Key, \_Tp, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
  - typedef \_Hashtable::value\_type [value\\_type](#)
  - typedef \_Hashtable::mapped\_type [mapped\\_type](#)
  - typedef \_Hashtable::hasher [hasher](#)
  - typedef \_Hashtable::key\_equal [key\\_equal](#)
  - typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- 
- typedef \_Hashtable::pointer [pointer](#)
  - typedef \_Hashtable::const\_pointer [const\\_pointer](#)
  - typedef \_Hashtable::reference [reference](#)
  - typedef \_Hashtable::const\_reference [const\\_reference](#)
  - typedef \_Hashtable::iterator [iterator](#)
  - typedef \_Hashtable::const\_iterator [const\\_iterator](#)
  - typedef \_Hashtable::local\_iterator [local\\_iterator](#)
  - typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
  - typedef \_Hashtable::size\_type [size\\_type](#)
  - typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_multimap](#) ()=default
- [unordered\\_multimap](#) (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[unordered\\_multimap](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_multimap](#) (const unordered\_multimap &)=default
- [unordered\\_multimap](#) (unordered\_multimap &&)=default
- [unordered\\_multimap](#) (const allocator\_type &\_\_a)
- **unordered\_multimap** (const unordered\_multimap &\_\_ummap, const allocator\_type &\_\_a)
- **unordered\_multimap** (unordered\_multimap &&\_\_ummap, const allocator\_type &\_\_a)
- [unordered\\_multimap](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)

- **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
  - **unordered\_multimap** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
  - [iterator](#) **begin** () noexcept
  - [local\\_iterator](#) **begin** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **bucket** (const [key\\_type](#) &\_\_key) const
  - [size\\_type](#) **bucket\_count** () const noexcept
  - [size\\_type](#) **bucket\_size** ([size\\_type](#) \_\_n) const
  - void **clear** () noexcept
  - [size\\_type](#) **count** (const [key\\_type](#) &\_\_x) const
  - template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
[iterator](#) **emplace\_hint** (const [iterator](#) \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - [iterator](#) **end** () noexcept
  - [local\\_iterator](#) **end** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
  - [iterator](#) **erase** (const [iterator](#) \_\_first, const [iterator](#) \_\_last)
  - [allocator\\_type](#) **get\_allocator** () const noexcept
  - [hasher](#) **hash\_function** () const
  - template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [key\\_equal](#) **key\_eq** () const
  - float **load\_factor** () const noexcept
  - [size\\_type](#) **max\_bucket\_count** () const noexcept
  - float **max\_load\_factor** () const noexcept
  - void **max\_load\_factor** (float \_\_z)
  - [size\\_type](#) **max\_size** () const noexcept
  - [unordered\\_multimap](#) & **operator=** (const [unordered\\_multimap](#) &)=default
  - [unordered\\_multimap](#) & **operator=** ([unordered\\_multimap](#) &&)=default
  - [unordered\\_multimap](#) & **operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - void **rehash** ([size\\_type](#) \_\_n)
  - void **reserve** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **size** () const noexcept
  - void **swap** ([unordered\\_multimap](#) &\_\_x) noexcept(noexcept(\_M\_h.swap(\_\_x.\_M\_h)))
- 
- [const\\_iterator](#) **begin** () const noexcept
  - [const\\_iterator](#) **cbegin** () const noexcept
- 
- [const\\_iterator](#) **end** () const noexcept
  - [const\\_iterator](#) **cend** () const noexcept

- [iterator insert](#) (const [value\\_type](#) &\_\_x)
- [iterator insert](#) ([value\\_type](#) &&\_\_x)
- [template<typename \\_Pair > \\_\\_enable\\_if\\_t< is\\_constructible< value\\_type, \\_Pair && >::value, iterator > insert](#) (\_Pair &&\_\_x)
- [iterator insert](#) (const\_iterator \_\_hint, const [value\\_type](#) &\_\_x)
- [iterator insert](#) (const\_iterator \_\_hint, [value\\_type](#) &&\_\_x)
- [template<typename \\_Pair > \\_\\_enable\\_if\\_t< is\\_constructible< value\\_type, \\_Pair && >::value, iterator > insert](#) (const\_iterator \_\_hint, \_Pair &&\_\_x)
- [iterator erase](#) (const\_iterator \_\_position)
- [iterator erase](#) (iterator \_\_position)
- [iterator find](#) (const [key\\_type](#) &\_\_x)
- [const\\_iterator find](#) (const [key\\_type](#) &\_\_x) const
- [std::pair< iterator, iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x)
- [std::pair< const\\_iterator, const\\_iterator > equal\\_range](#) (const [key\\_type](#) &\_\_x) const
- [const\\_local\\_iterator begin](#) (size\_type \_\_n) const
- [const\\_local\\_iterator cbegin](#) (size\_type \_\_n) const
- [const\\_local\\_iterator end](#) (size\_type \_\_n) const
- [const\\_local\\_iterator cend](#) (size\_type \_\_n) const

#### Friends

- [template<typename \\_Key1, typename \\_Tp1, typename \\_Hash1, typename \\_Pred1, typename \\_Alloc1 > bool operator==](#) (const [unordered\\_multimap](#)< \_Key1, \_Tp1, \_Hash1, \_Pred1, \_Alloc1 > &, const [unordered\\_multimap](#)< \_Key1, \_Tp1, \_Hash1, \_Pred1, \_Alloc1 > &)

#### 4.1036.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc =
allocator<std::pair<const _Key, _Tp>>>>
class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

### Template Parameters

|                     |                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------|
| <code>_Key</code>   | Type of key objects.                                                                              |
| <code>_Tp</code>    | Type of mapped objects.                                                                           |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .                       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> .                 |
| <code>_Alloc</code> | Allocator type, defaults to <code>std::allocator&lt;std::pair&lt;const _Key, _Tp&gt;&gt;</code> . |

Meets the requirements of a `container`, and `unordered associative container`

The resulting value type of the container is `std::pair<const _Key, _Tp>`.

Base is `_Hashtable`, dispatched at compile time via template alias `__ummap_hashtable`.

Definition at line 73 of file `unordered_map.h`.

## 4.1036.2 Member Typedef Documentation

### 4.1036.2.1 `allocator_type`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::allocator_type
```

Public typedefs.

Definition at line 1263 of file `unordered_map.h`.

### 4.1036.2.2 `const_iterator`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::const_iterator
```

Iterator-related typedefs.

Definition at line 1273 of file `unordered_map.h`.

#### 4.1036.2.3 const\_local\_iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::const_local_iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

Definition at line 1275 of file unordered\_map.h.

#### 4.1036.2.4 const\_pointer

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::const_pointer std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵
::const_pointer
```

Iterator-related typedefs.

Definition at line 1269 of file unordered\_map.h.

#### 4.1036.2.5 const\_reference

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::const_reference std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵
::const_reference
```

Iterator-related typedefs.

Definition at line 1271 of file unordered\_map.h.

#### 4.1036.2.6 difference\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::difference_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >↵
::difference_type
```

Iterator-related typedefs.

Definition at line 1277 of file unordered\_map.h.

#### 4.1036.2.7 hasher

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 1261 of file unordered\_map.h.

#### 4.1036.2.8 iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 1272 of file unordered\_map.h.

#### 4.1036.2.9 key\_equal

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 1262 of file unordered\_map.h.

#### 4.1036.2.10 key\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 1258 of file unordered\_map.h.



#### 4.1036.2.11 local\_iterator

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::local_iterator
```

Iterator-related typedefs.

Definition at line 1274 of file unordered\_map.h.

#### 4.1036.2.12 mapped\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↵
::mapped_type
```

Public typedefs.

Definition at line 1260 of file unordered\_map.h.

#### 4.1036.2.13 pointer

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 1268 of file unordered\_map.h.

#### 4.1036.2.14 reference

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 1270 of file unordered\_map.h.

#### 4.1036.2.15 size\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 1276 of file unordered\_map.h.

#### 4.1036.2.16 value\_type

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 1259 of file unordered\_map.h.

### 4.1036.3 Constructor & Destructor Documentation

#### 4.1036.3.1 unordered\_multimap() [1/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap () [default]
```

Default constructor.

#### 4.1036.3.2 unordered\_multimap() [2/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Definition at line 1297 of file `unordered_map.h`.

## 4.1036.3.3 unordered\_multimap() [3/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator >
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_multimap` from a range.

## Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_multimap` consisting of copies of the elements from `[__first,__last)`. This is linear in N (where N is `distance(__first,__last)`).

Definition at line 1318 of file `unordered_map.h`.

## 4.1036.3.4 unordered\_multimap() [4/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

**4.1036.3.5 unordered\_multimap()** [5/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

**4.1036.3.6 unordered\_multimap()** [6/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_multimap with no elements.

**Parameters**

|                                   |                      |
|-----------------------------------|----------------------|
| <code>↵</code><br><code>_a</code> | An allocator object. |
|-----------------------------------|----------------------|

Definition at line 1337 of file unordered\_map.h.

**4.1036.3.7 unordered\_multimap()** [7/7]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_multimap from an initializer\_list.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_multimap consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 1372 of file unordered\_map.h.

#### 4.1036.4 Member Function Documentation

##### 4.1036.4.1 begin() [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered\_multimap.

Definition at line 1470 of file unordered\_map.h.

##### 4.1036.4.2 begin() [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 1479 of file unordered\_map.h.

##### 4.1036.4.3 begin() [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read/write local iterator.

Definition at line 1895 of file unordered\_map.h.

**4.1036.4.4 begin()** [4/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                        |                   |
|------------------------|-------------------|
| <code>↵<br/>__n</code> | The bucket index. |
|------------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1906 of file unordered\_map.h.

**4.1036.4.5 bucket\_count()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const
[inline], [noexcept]
```

Returns the number of buckets of the unordered\_multimap.

Definition at line 1862 of file unordered\_map.h.

**4.1036.4.6 cbegin()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multimap.

Definition at line 1483 of file unordered\_map.h.

## 4.1036.4.7 cbegin() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>_↵</code>  | The bucket index. |
| <code>__n</code> |                   |

## Returns

A read-only local iterator.

Definition at line 1910 of file unordered\_map.h.

## 4.1036.4.8 cend() [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multimap.

Definition at line 1505 of file unordered\_map.h.

## 4.1036.4.9 cend() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>_↵</code>  | The bucket index. |
| <code>__n</code> |                   |

**Returns**

A read-only local iterator.

Definition at line 1936 of file unordered\_map.h.

**4.1036.4.10 clear()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1734 of file unordered\_map.h.

**4.1036.4.11 count()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                 |               |
|-----------------|---------------|
| <code>_↔</code> | Key to count. |
| <code>_X</code> |               |

**Returns**

Number of elements with specified key.

Definition at line 1828 of file unordered\_map.h.

**4.1036.4.12 emplace()**

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the unordered\_multimap.



## Parameters

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the `unordered_multimap`.

Insertion requires amortized constant time.

Definition at line 1528 of file `unordered_map.h`.

4.1036.4.13 `emplace_hint()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to build and insert a `std::pair` into the `unordered_multimap`.

## Parameters

|                     |                                                                                                                                                        |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the pair should be inserted.                                                                             |
| <code>__args</code> | Arguments used to generate a new pair instance (see <code>std::piecewise_construct</code> for passing arguments to each part of the pair constructor). |

## Returns

An iterator that points to the element with key of the `std::pair` built from `__args`.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1555 of file `unordered_map.h`.

**4.1036.4.14** `empty()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the `unordered_multimap` is empty.

Definition at line 1450 of file `unordered_map.h`.

**4.1036.4.15** `end()` [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the `unordered_multimap`.

Definition at line 1492 of file `unordered_map.h`.

**4.1036.4.16** `end()` [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multimap`.

Definition at line 1501 of file `unordered_map.h`.

**4.1036.4.17** `end()` [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read/write local iterator.

Definition at line 1921 of file unordered\_map.h.

**4.1036.4.18 end()** [ 4 / 4 ]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                       |                   |
|-----------------------|-------------------|
| <b>↵</b><br><b>_n</b> | The bucket index. |
|-----------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1932 of file unordered\_map.h.

**4.1036.4.19 equal\_range()** [ 1 / 2 ]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<iterator, iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_↵
_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                       |                    |
|-----------------------|--------------------|
| <b>↵</b><br><b>_x</b> | Key to be located. |
|-----------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1850 of file unordered\_map.h.

#### 4.1036.4.20 equal\_range() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair<const_iterator, const_iterator> std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _↵
Alloc >::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

##### Parameters

|                  |                    |
|------------------|--------------------|
| <code>↵</code>   | Key to be located. |
| <code>__x</code> |                    |

##### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1854 of file unordered\_map.h.

#### 4.1036.4.21 erase() [1/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_multimap.

##### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1684 of file unordered\_map.h.

#### 4.1036.4.22 erase() [2/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an unordered\_multimap.

##### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

##### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1689 of file unordered\_map.h.

#### 4.1036.4.23 erase() [3/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

##### Parameters

|                  |                               |
|------------------|-------------------------------|
| <code>__x</code> | Key of elements to be erased. |
|------------------|-------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multimap`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1705 of file `unordered_map.h`.

**4.1036.4.24 erase()** [4/4]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multimap`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multimap`. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1724 of file `unordered_map.h`.

**4.1036.4.25 find()** [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>_X</code>   |                    |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1814 of file `unordered_map.h`.

## 4.1036.4.26 find() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an `unordered_multimap`.

## Parameters

|                   |                    |
|-------------------|--------------------|
| <code>_Key</code> | Key to be located. |
| <code>_X</code>   |                    |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1818 of file `unordered_map.h`.

## 4.1036.4.27 get\_allocator()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>,
typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator ()
const [inline], [noexcept]
```

Returns the allocator object used by the `unordered_multimap`.

Definition at line 1443 of file `unordered_map.h`.

**4.1036.4.28** `hash_function()`

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the `unordered_multimap` was constructed.

Definition at line 1790 of file `unordered_map.h`.

**4.1036.4.29** `insert()` [1/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

|                        |                                                                                   |
|------------------------|-----------------------------------------------------------------------------------|
| <code>↵<br/>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------|-----------------------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1569 of file `unordered_map.h`.

**4.1036.4.30** `insert()` [2/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

|                        |                                                                                   |
|------------------------|-----------------------------------------------------------------------------------|
| <code>↵<br/>__x</code> | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |
|------------------------|-----------------------------------------------------------------------------------|



**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1573 of file unordered\_map.h.

References std::move().

**4.1036.4.31 insert()** [3/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                       |                                                                      |
|-----------------------|----------------------------------------------------------------------|
| <b>↔</b><br><b>_x</b> | Pair to be inserted (see std::make_pair for easy creation of pairs). |
|-----------------------|----------------------------------------------------------------------|

**Returns**

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

Definition at line 1578 of file unordered\_map.h.

**4.1036.4.32 insert()** [4/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
iterator std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1603 of file `unordered_map.h`.

**4.1036.4.33 insert()** [5/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Inserts a `std::pair` into the `unordered_multimap`.

**Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted.        |
| <code>__x</code>    | Pair to be inserted (see <code>std::make_pair</code> for easy creation of pairs). |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1609 of file `unordered_map.h`.

References `std::move()`.

## 4.1036.4.34 insert() [6/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _Pair >
__enable_if_t<is_constructible<value_type, _Pair&&>::value, iterator> std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 _Pair && __x) [inline]
```

Inserts a std::pair into the unordered\_multimap.

## Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the pair should be inserted. |
| <code>__x</code>    | Pair to be inserted (see std::make_pair for easy creation of pairs).       |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↵associative.insert_hints) for more on *hinting*.

Insertion requires amortized constant time.

Definition at line 1614 of file unordered\_map.h.

## 4.1036.4.35 insert() [7/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
template<typename _InputIterator >
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

## Parameters

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 1629 of file unordered\_map.h.

#### 4.1036.4.36 insert() [8/8]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the unordered\_multimap.

##### Parameters

|    |                                                                 |
|----|-----------------------------------------------------------------|
| ↵  | A std::initializer_list<value_type> of elements to be inserted. |
| _↵ |                                                                 |
| ↵  |                                                                 |
| _↵ |                                                                 |
| /  |                                                                 |

Complexity similar to that of the range constructor.

Definition at line 1641 of file unordered\_map.h.

#### 4.1036.4.37 key\_eq()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_multimap was constructed.

Definition at line 1796 of file unordered\_map.h.

#### 4.1036.4.38 load\_factor()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1944 of file unordered\_map.h.

**4.1036.4.39** max\_bucket\_count()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multimap.

Definition at line 1867 of file unordered\_map.h.

**4.1036.4.40** max\_load\_factor() [1/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
float std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multimap tries to keep the load factor less than or equal to.

Definition at line 1950 of file unordered\_map.h.

**4.1036.4.41** max\_load\_factor() [2/2]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the unordered\_multimap maximum load factor.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <b>_↔<br/>_z</b> | The new maximum load factor. |
|------------------|------------------------------|

Definition at line 1958 of file unordered\_map.h.

**4.1036.4.42** max\_size()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↔
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>>
size_type std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multimap.

Definition at line 1460 of file unordered\_map.h.

#### 4.1036.4.43 operator=() [1/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

#### 4.1036.4.44 operator=() [2/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

#### 4.1036.4.45 operator=() [3/3]

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap& std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_multimap list assignment operator.

##### Parameters

|    |                      |
|----|----------------------|
| ↵  | An initializer_list. |
| _↵ |                      |
| ↵  |                      |
| _↵ |                      |
| /  |                      |

This function fills an unordered\_multimap with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_multimap and that the resulting unordered\_multimap's size is the same as the number of elements assigned.

Definition at line 1435 of file unordered\_map.h.

#### 4.1036.4.46 rehash()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_multimap.

##### Parameters

|                        |                            |
|------------------------|----------------------------|
| <code>↵<br/>__n</code> | The new number of buckets. |
|------------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multimap maximum load factor.

Definition at line 1969 of file unordered\_map.h.

#### 4.1036.4.47 reserve()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_multimap for a specified number of elements.

##### Parameters

|                        |                              |
|------------------------|------------------------------|
| <code>↵<br/>__n</code> | Number of elements required. |
|------------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

Definition at line 1980 of file unordered\_map.h.

#### 4.1036.4.48 size()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const [inline],
[noexcept]
```

Returns the size of the unordered\_multimap.

Definition at line 1455 of file unordered\_map.h.

#### 4.1036.4.49 swap()

```
template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = equal_↵
to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
 unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_multimap.

#### Parameters

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>↵</code>   | An unordered_multimap of the same element and allocator types. |
| <code>__X</code> |                                                                |

This exchanges the elements between two unordered\_multimap in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(m1,m2)` will feed to this function.

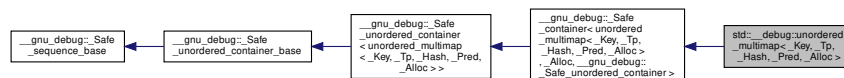
Definition at line 1748 of file unordered\_map.h.

The documentation for this class was generated from the following file:

- [unordered\\_map.h](#)

#### 4.1037 std::\_\_debug::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inheritance diagram for `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`:





## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::\_Safe\_iterator<\_Base\_const\_iterator, unordered\_multimap > **const\_iterator**
- typedef \_\_gnu\_debug::\_Safe\_local\_iterator<\_Base\_const\_local\_iterator, unordered\_multimap > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef \_\_gnu\_debug::\_Safe\_iterator<\_Base\_iterator, unordered\_multimap > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_\_gnu\_debug::\_Safe\_local\_iterator<\_Base\_local\_iterator, unordered\_multimap > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (const unordered\_multimap &)=default
- **unordered\_multimap** (const \_Base &\_\_x)
- **unordered\_multimap** (unordered\_multimap &&)=default
- **unordered\_multimap** (const allocator\_type &\_\_a)
- **unordered\_multimap** (const unordered\_multimap &\_\_umap, const allocator\_type &\_\_a)
- **unordered\_multimap** (unordered\_multimap &&\_\_umap, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multimap** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multimap** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multimap** (initializer\_list<value\_type> \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **\_Base & \_M\_base** () noexcept
- const **\_Base & \_M\_base** () const noexcept
- void **\_M\_swap** (\_Safe\_container &\_\_x) noexcept
- **iterator begin** () noexcept
- const **iterator begin** () const noexcept
- **local\_iterator begin** (size\_type \_\_b)
- const **local\_iterator begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- const **iterator cbegin** () const noexcept
- const **local\_iterator cbegin** (size\_type \_\_b) const
- const **iterator cend** () const noexcept
- const **local\_iterator cend** (size\_type \_\_b) const

- void **clear** () noexcept
- template<typename... \_Args>  
iterator **emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
iterator **emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- local\_iterator **end** (size\_type \_\_b)
- const\_local\_iterator **end** (size\_type \_\_b) const
- std::pair< iterator, iterator > **equal\_range** (const key\_type &\_\_key)
- std::pair< const\_iterator, const\_iterator > **equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- iterator **erase** (const\_iterator \_\_it)
- iterator **erase** (iterator \_\_it)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- iterator **find** (const key\_type &\_\_key)
- const\_iterator **find** (const key\_type &\_\_key) const
- iterator **insert** (const value\_type &\_\_obj)
- iterator **insert** (value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_hint, const value\_type &\_\_obj)
- iterator **insert** (const\_iterator \_\_hint, value\_type &&\_\_x)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (\_Pair &&\_\_obj)
- template<typename \_Pair, typename = typename std::enable\_if<std::is\_constructible<value\_type, \_Pair&&>::value>::type>  
iterator **insert** (const\_iterator \_\_hint, \_Pair &&\_\_obj)
- void **insert** (std::initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- unordered\_multimap & **operator=** (const unordered\_multimap &)=default
- unordered\_multimap & **operator=** (unordered\_multimap &&)=default
- unordered\_multimap & **operator=** (initializer\_list< value\_type > \_\_l)
- void **swap** (unordered\_multimap &\_\_x) noexcept(noexcept(declval< \_Base & >().swap(\_\_x)))

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_const\_local\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- \_Safe\_iterator\_base \* **\_M\_local\_iterators**
- unsigned int **\_M\_version**

## Protected Member Functions

- void [\\_M\\_detach\\_all](#) ()
- void [\\_M\\_detach\\_singular](#) ()
- [\\_\\_gnu\\_cxx::\\_\\_mutex](#) & [\\_M\\_get\\_mutex](#) () throw ()
- void [\\_M\\_invalidate\\_all](#) ()
- void [\\_M\\_invalidate\\_all](#) () const
- void [\\_M\\_invalidate\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_local\\_if](#) (\_Predicate \_\_pred)
- void [\\_M\\_invalidate\\_locals](#) ()
- void [\\_M\\_revalidate\\_singular](#) ()
- [\\_Safe\\_container](#) & [\\_M\\_safe](#) () noexcept
- void [\\_M\\_swap](#) (\_Safe\_unordered\_container\_base &\_\_x) noexcept
- void [\\_M\\_swap](#) (\_Safe\_sequence\_base &\_\_x) noexcept

## Friends

- template<typename \_ItT, typename \_SeqT, typename \_CatT >  
class [::\\_\\_gnu\\_debug::Safe\\_iterator](#)
- template<typename \_ItT, typename \_SeqT >  
class [::\\_\\_gnu\\_debug::Safe\\_local\\_iterator](#)

## 4.1037.1 Detailed Description

```
template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename
_Alloc = std::allocator<std::pair<const _Key, _Tp>>>
class std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>
```

Class std::unordered\_multimap with safety/checking/debug instrumentation.

Definition at line 44 of file debug/unordered\_map.

## 4.1037.2 Member Function Documentation

4.1037.2.1 [\\_M\\_detach\\_all\(\)](#)

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### 4.1037.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

##### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 4.1037.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1037.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.1037.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> >::_M_invalidate_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

## 4.1037.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
> >::_M_invalidate_local_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all local iterators  $x$  that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

## 4.1037.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.1037.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.1037.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.1037.3 Member Data Documentation

## 4.1037.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1037.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 4.1037.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1037.3.4 `_M_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

#### 4.1037.3.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_map](#)

## 4.1038 std::unordered\_multiset&lt; \_Value, \_Hash, \_Pred, \_Alloc &gt; Class Template Reference

## Public Types

- typedef \_Hashtable::key\_type [key\\_type](#)
  - typedef \_Hashtable::value\_type [value\\_type](#)
  - typedef \_Hashtable::hasher [hasher](#)
  - typedef \_Hashtable::key\_equal [key\\_equal](#)
  - typedef \_Hashtable::allocator\_type [allocator\\_type](#)
- 
- typedef \_Hashtable::pointer [pointer](#)
  - typedef \_Hashtable::const\_pointer [const\\_pointer](#)
  - typedef \_Hashtable::reference [reference](#)
  - typedef \_Hashtable::const\_reference [const\\_reference](#)
  - typedef \_Hashtable::iterator [iterator](#)
  - typedef \_Hashtable::const\_iterator [const\\_iterator](#)
  - typedef \_Hashtable::local\_iterator [local\\_iterator](#)
  - typedef \_Hashtable::const\_local\_iterator [const\\_local\\_iterator](#)
  - typedef \_Hashtable::size\_type [size\\_type](#)
  - typedef \_Hashtable::difference\_type [difference\\_type](#)

## Public Member Functions

- [unordered\\_multiset](#) ()=default
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &)=default
- [unordered\\_multiset](#) ([unordered\\_multiset](#) &&)=default
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n=0, const [hasher](#) &\_\_hf=[hasher](#)(), const [key\\_equal](#) &\_\_eq=[key\\_equal](#)(), const [allocator\\_type](#) &\_\_a=[allocator\\_type](#)())
- [unordered\\_multiset](#) (const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) (const [unordered\\_multiset](#) &\_\_umset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([unordered\\_multiset](#) &&\_\_umset, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_multiset](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
- [unordered\\_multiset](#) ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [allocator\\_type](#) &\_\_a)

- **unordered\_multiset** ([initializer\\_list](#)< [value\\_type](#) > \_\_l, [size\\_type](#) \_\_n, const [hasher](#) &\_\_hf, const [allocator\\_type](#) &\_\_a)
  - [size\\_type](#) **bucket** (const [key\\_type](#) &\_\_key) const
  - [size\\_type](#) **bucket\_count** () const noexcept
  - [size\\_type](#) **bucket\_size** ([size\\_type](#) \_\_n) const
  - [const\\_iterator](#) **cbegin** () const noexcept
  - [const\\_iterator](#) **cend** () const noexcept
  - void **clear** () noexcept
  - [size\\_type](#) **count** (const [key\\_type](#) &\_\_x) const
  - template<typename... \_Args>  
[iterator](#) **emplace** (\_Args &&... \_\_args)
  - template<typename... \_Args>  
[iterator](#) **emplace\_hint** ([const\\_iterator](#) \_\_pos, \_Args &&... \_\_args)
  - bool **empty** () const noexcept
  - [size\\_type](#) **erase** (const [key\\_type](#) &\_\_x)
  - [iterator](#) **erase** ([const\\_iterator](#) \_\_first, [const\\_iterator](#) \_\_last)
  - [allocator\\_type](#) **get\_allocator** () const noexcept
  - [hasher](#) **hash\_function** () const
  - template<typename \_InputIterator >  
void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
  - void **insert** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - [key\\_equal](#) **key\_eq** () const
  - float **load\_factor** () const noexcept
  - [size\\_type](#) **max\_bucket\_count** () const noexcept
  - float **max\_load\_factor** () const noexcept
  - void **max\_load\_factor** (float \_\_z)
  - [size\\_type](#) **max\_size** () const noexcept
  - [unordered\\_multiset](#) & **operator=** (const [unordered\\_multiset](#) &)=default
  - [unordered\\_multiset](#) & **operator=** ([unordered\\_multiset](#) &&)=default
  - [unordered\\_multiset](#) & **operator=** ([initializer\\_list](#)< [value\\_type](#) > \_\_l)
  - void **rehash** ([size\\_type](#) \_\_n)
  - void **reserve** ([size\\_type](#) \_\_n)
  - [size\\_type](#) **size** () const noexcept
  - void **swap** ([unordered\\_multiset](#) &\_\_x) noexcept(noexcept(\_\_M\_h.swap(\_\_x.\_M\_h)))
- 
- [iterator](#) **begin** () noexcept
  - [const\\_iterator](#) **begin** () const noexcept
- 
- [iterator](#) **end** () noexcept
  - [const\\_iterator](#) **end** () const noexcept
- 
- [iterator](#) **insert** (const [value\\_type](#) &\_\_x)



- `iterator insert (value_type &&__x)`
- `iterator insert (const_iterator __hint, const value_type &__x)`
- `iterator insert (const_iterator __hint, value_type &&__x)`
- `iterator erase (const_iterator __position)`
- `iterator erase (iterator __position)`
- `iterator find (const key_type &__x)`
- `const_iterator find (const key_type &__x) const`
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
- `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- `local_iterator begin (size_type __n)`
- `const_local_iterator begin (size_type __n) const`
- `const_local_iterator cbegin (size_type __n) const`
- `local_iterator end (size_type __n)`
- `const_local_iterator end (size_type __n) const`
- `const_local_iterator cend (size_type __n) const`

#### Friends

- `template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1 >  
bool operator==(const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)`

#### 4.1038.1 Detailed Description

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc =
allocator<_Value>>
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

**Template Parameters**

|                     |                                                                                   |
|---------------------|-----------------------------------------------------------------------------------|
| <code>_Value</code> | Type of key objects.                                                              |
| <code>_Hash</code>  | Hashing function object type, defaults to <code>hash&lt;_Value&gt;</code> .       |
| <code>_Pred</code>  | Predicate function object type, defaults to <code>equal_to&lt;_Value&gt;</code> . |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Key&gt;</code> .                  |

Meets the requirements of a `container`, and `unordered associative container`

Base is `_Hashtable`, dispatched at compile time via template alias `__umset_hashtable`.

Definition at line 70 of file `unordered_set.h`.

**4.1038.2 Member Typedef Documentation****4.1038.2.1 allocator\_type**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::allocator_type
```

Public typedefs.

Definition at line 925 of file `unordered_set.h`.

**4.1038.2.2 const\_iterator**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::const_iterator
```

Iterator-related typedefs.

Definition at line 935 of file `unordered_set.h`.

#### 4.1038.2.3 const\_local\_iterator

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

Definition at line 937 of file unordered\_set.h.

#### 4.1038.2.4 const\_pointer

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 931 of file unordered\_set.h.

#### 4.1038.2.5 const\_reference

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::const_reference
```

Iterator-related typedefs.

Definition at line 933 of file unordered\_set.h.

#### 4.1038.2.6 difference\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::difference_type
```

Iterator-related typedefs.

Definition at line 939 of file unordered\_set.h.

#### 4.1038.2.7 hasher

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

Definition at line 923 of file `unordered_set.h`.

#### 4.1038.2.8 iterator

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

Definition at line 934 of file `unordered_set.h`.

#### 4.1038.2.9 key\_equal

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

Definition at line 924 of file `unordered_set.h`.

#### 4.1038.2.10 key\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 921 of file `unordered_set.h`.

#### 4.1038.2.11 `local_iterator`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↵
::local_iterator
```

Iterator-related typedefs.

Definition at line 936 of file `unordered_set.h`.

#### 4.1038.2.12 `pointer`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 930 of file `unordered_set.h`.

#### 4.1038.2.13 `reference`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 932 of file `unordered_set.h`.

#### 4.1038.2.14 `size_type`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 938 of file `unordered_set.h`.

#### 4.1038.2.15 value\_type

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 922 of file unordered\_set.h.

### 4.1038.3 Constructor & Destructor Documentation

#### 4.1038.3.1 unordered\_multiset() [1/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset () [default]
```

Default constructor.

#### 4.1038.3.2 unordered\_multiset() [2/7]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

##### Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Definition at line 959 of file unordered\_set.h.

## 4.1038.3.3 unordered\_multiset() [3/7]

```

template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]

```

Builds an unordered\_multiset from a range.

## Parameters

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eq1</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an unordered\_multiset consisting of copies of the elements from [`__first`,`__last`). This is linear in N (where N is distance(`__first`,`__last`)).

Definition at line 980 of file unordered\_set.h.

## 4.1038.3.4 unordered\_multiset() [4/7]

```

template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &) [default]

```

Copy constructor.

## 4.1038.3.5 unordered\_multiset() [5/7]

```

template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > &&) [default]

```

Move constructor.

## 4.1038.3.6 unordered\_multiset() [6/7]

```

template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]

```

Builds an unordered\_multiset from an initializer\_list.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an unordered\_multiset consisting of copies of the elements in the list. This is linear in N (where N is `__l.size()`).

Definition at line 1005 of file unordered\_set.h.

## 4.1038.3.7 unordered\_multiset() [7/7]

```

template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
 const allocator_type & __a) [inline], [explicit]

```

Creates an unordered\_multiset with no elements.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Definition at line 1026 of file unordered\_set.h.

## 4.1038.4 Member Function Documentation



**4.1038.4.1** begin() [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 1133 of file unordered\_set.h.

**4.1038.4.2** begin() [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_multiset.

Definition at line 1137 of file unordered\_set.h.

**4.1038.4.3** begin() [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1530 of file unordered\_set.h.

**4.1038.4.4** `begin()` [4/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1534 of file `unordered_set.h`.

**4.1038.4.5** `bucket_count()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the `unordered_multiset`.

Definition at line 1496 of file `unordered_set.h`.

**4.1038.4.6** `cbegin()` [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_multiset`.

Definition at line 1160 of file `unordered_set.h`.

**4.1038.4.7** `cbegin()` [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

## Parameters

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

## Returns

A read-only local iterator.

Definition at line 1538 of file unordered\_set.h.

## 4.1038.4.8 cend() [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered\_multiset.

Definition at line 1168 of file unordered\_set.h.

## 4.1038.4.9 cend() [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

## Parameters

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

## Returns

A read-only local iterator.

Definition at line 1558 of file unordered\_set.h.

#### 4.1038.4.10 clear()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_multiset.

Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.


Definition at line 1369 of file unordered\_set.h.

#### 4.1038.4.11 count()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

##### Parameters

|                                                                                                      |                     |
|------------------------------------------------------------------------------------------------------|---------------------|
|  <code>__x</code> | Element to located. |
|------------------------------------------------------------------------------------------------------|---------------------|

##### Returns

Number of elements with specified key.

Definition at line 1462 of file unordered\_set.h.

#### 4.1038.4.12 emplace()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Builds and insert an element into the unordered\_multiset.

##### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 1182 of file unordered\_set.h.

**4.1038.4.13   emplace\_hint()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

**Returns**

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant time.

Definition at line 1204 of file unordered\_set.h.

**4.1038.4.14   empty()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the unordered\_multiset is empty.

Definition at line 1112 of file unordered\_set.h.

**4.1038.4.15** `end()` [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 1147 of file `unordered_set.h`.

**4.1038.4.16** `end()` [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_multiset`.

Definition at line 1151 of file `unordered_set.h`.

**4.1038.4.17** `end()` [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 1550 of file `unordered_set.h`.

**4.1038.4.18** end() [4/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                 |                   |
|-----------------|-------------------|
| <code>_↵</code> | The bucket index. |
| <code>_n</code> |                   |

**Returns**

A read-only local iterator.

Definition at line 1554 of file unordered\_set.h.

**4.1038.4.19** equal\_range() [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_↵
range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                 |                    |
|-----------------|--------------------|
| <code>_↵</code> | Key to be located. |
| <code>_x</code> |                    |

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1484 of file unordered\_set.h.

**4.1038.4.20** equal\_range() [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::pair<const_iterator, const_iterator> std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

#### Parameters

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

#### Returns

Pair of iterators that possibly points to the subsequence matching given key.

Definition at line 1488 of file unordered\_set.h.

#### 4.1038.4.21 erase() [1/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_multiset.

#### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

#### Returns

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an unordered\_multiset.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1315 of file unordered\_set.h.



## 4.1038.4.22 erase() [2/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an unordered\_multiset.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1320 of file `unordered_set.h`.

**4.1038.4.23 erase()** [3/4]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__x</code> | Key of element to be erased. |
|------------------|------------------------------|

**Returns**

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1338 of file `unordered_set.h`.

## 4.1038.4.24 erase() [ 4/4 ]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_multiset`.

**Parameters**

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

**Returns**

The iterator `__last`.

This function erases a sequence of elements from an `unordered_multiset`.

Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1358 of file `unordered_set.h`.

**4.1038.4.25 find()** [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an `unordered_multiset`.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__x</code> | Element to be located. |
|------------------|------------------------|

**Returns**

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

Definition at line 1448 of file `unordered_set.h`.

**4.1038.4.26 find()** [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an `unordered_multiset`.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <code>_↵</code> | Element to be located. |
| <code>_X</code> |                        |

## Returns

Iterator pointing to sought-after element, or `end()` if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 1452 of file `unordered_set.h`.

4.1038.4.27 `get_allocator()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the `unordered_multiset`.

Definition at line 1105 of file `unordered_set.h`.

4.1038.4.28 `hash_function()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the `unordered_multiset` was constructed.



Definition at line 1424 of file `unordered_set.h`.

4.1038.4.29 `insert()` [1/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Inserts an element into the `unordered_multiset`.

**Parameters**

|                                                                                   |                         |
|-----------------------------------------------------------------------------------|-------------------------|
|  | Element to be inserted. |
|  |                         |

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.



Definition at line 1216 of file unordered\_set.h.

**4.1038.4.30 insert()** [2/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

**Parameters**

|                                                                                     |                         |
|-------------------------------------------------------------------------------------|-------------------------|
|  | Element to be inserted. |
|  |                         |

**Returns**

An iterator that points to the inserted element.

Insertion requires amortized constant time.

Definition at line 1220 of file unordered\_set.h.

References std::move().

**4.1038.4.31 insert()** [3/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Inserts an element into the unordered\_multiset.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 1242 of file unordered\_set.h.

## 4.1038.4.32 insert() [4/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Inserts an element into the unordered\_multiset.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

## Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 1246 of file unordered\_set.h.

References std::move().

**4.1038.4.33 insert()** [5/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that inserts a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 1260 of file unordered\_set.h.

**4.1038.4.34 insert()** [6/6]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Inserts a list of elements into the unordered\_multiset.

**Parameters**

|                          |                                                                 |
|--------------------------|-----------------------------------------------------------------|
| <pre>↔ __↔ ↔ __↔ /</pre> | A std::initializer_list<value_type> of elements to be inserted. |
|--------------------------|-----------------------------------------------------------------|

Complexity similar to that of the range constructor.

Definition at line 1271 of file unordered\_set.h.

**4.1038.4.35 key\_eq()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```



Returns the key comparison object with which the unordered\_multiset was constructed.

Definition at line 1430 of file unordered\_set.h.

#### 4.1038.4.36 load\_factor()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline],
[noexcept]
```

Returns the average number of elements per bucket.

Definition at line 1566 of file unordered\_set.h.

#### 4.1038.4.37 max\_bucket\_count()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered\_multiset.

Definition at line 1501 of file unordered\_set.h.

#### 4.1038.4.38 max\_load\_factor() [1/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_multiset tries to keep the load factor less than or equal to.


Definition at line 1572 of file unordered\_set.h.

#### 4.1038.4.39 max\_load\_factor() [2/2]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the unordered\_multiset maximum load factor.

**Parameters**

|                                                                                                   |                              |
|---------------------------------------------------------------------------------------------------|------------------------------|
| <a href="#"></a> | The new maximum load factor. |
| <a href="#">_Z</a>                                                                                |                              |

Definition at line 1580 of file unordered\_set.h.

**4.1038.4.40 max\_size()**

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline],
[noexcept]
```

Returns the maximum size of the unordered\_multiset.

Definition at line 1122 of file unordered\_set.h.

**4.1038.4.41 operator=()** [1/3]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

**4.1038.4.42 operator=()** [2/3]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > &&) [default]
```






Move assignment operator.

**4.1038.4.43 operator=()** [3/3]

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset& std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_multiset list assignment operator.

## Parameters

|                                                                                   |                      |
|-----------------------------------------------------------------------------------|----------------------|
|  | An initializer_list. |
|  |                      |
|  |                      |
|  |                      |
|  |                      |

This function fills an unordered\_multiset with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_multiset and that the resulting unordered\_multiset's size is the same as the number of elements assigned.



Definition at line 1097 of file unordered\_set.h.

## 4.1038.4.44 rehash()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_multiset.

## Parameters

|                                                                                                                                                                            |                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| <br> | The new number of buckets. |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_multiset maximum load factor.



Definition at line 1591 of file unordered\_set.h.

## 4.1038.4.45 reserve()

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_multiset for a specified number of elements.

## Parameters

|                                                                                                                                                                            |                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|
| <br> | Number of elements required. |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------|

Same as `rehash(ceil(n / max_load_factor()))`.

Definition at line 1602 of file `unordered_set.h`.

#### 4.1038.4.46 `size()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size () const [inline],
[noexcept]
```

Returns the size of the `unordered_multiset`.

Definition at line 1117 of file `unordered_set.h`.

#### 4.1038.4.47 `swap()`

```
template<typename _Value , typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
 unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another `unordered_multiset`.

##### Parameters

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| <code>__x</code> | An <code>unordered_multiset</code> of the same element and allocator types. |
|------------------|-----------------------------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global `std::swap()` function is specialized such that `std::swap(s1,s2)` will feed to this function.

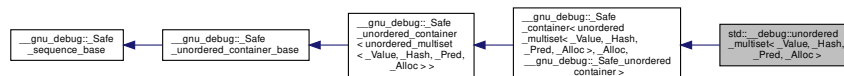
Definition at line 1382 of file `unordered_set.h`.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

#### 4.1039 `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >` Class Template Reference

Inheritance diagram for `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`:



## Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::\_\_Safe\_iterator< \_Base\_const\_iterator, unordered\_multiset > **const\_iterator**
- typedef \_\_gnu\_debug::\_\_Safe\_local\_iterator< \_Base\_const\_local\_iterator, unordered\_multiset > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef \_\_gnu\_debug::\_\_Safe\_iterator< \_Base\_iterator, unordered\_multiset > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_\_gnu\_debug::\_\_Safe\_local\_iterator< \_Base\_local\_iterator, unordered\_multiset > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (const unordered\_multiset &)=default
- **unordered\_multiset** (const \_Base &\_\_x)
- **unordered\_multiset** (unordered\_multiset &&)=default
- **unordered\_multiset** (const allocator\_type &\_\_a)
- **unordered\_multiset** (const unordered\_multiset &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** (unordered\_multiset &&\_\_uset, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_multiset** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_multiset** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_multiset** (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- \_Base & **\_M\_base** () noexcept
- const \_Base & **\_M\_base** () const noexcept
- void **\_M\_swap** (\_Safe\_container &\_\_x) noexcept
- **iterator begin** () noexcept
- **const\_iterator begin** () const noexcept
- **local\_iterator begin** (size\_type \_\_b)
- **const\_local\_iterator begin** (size\_type \_\_b) const
- size\_type **bucket\_size** (size\_type \_\_b) const
- **const\_iterator cbegin** () const noexcept
- **const\_local\_iterator cbegin** (size\_type \_\_b) const
- **const\_iterator cend** () const noexcept
- **const\_local\_iterator cend** (size\_type \_\_b) const

- void **clear** () noexcept
- template<typename... \_Args>  
    **iterator emplace** (\_Args &&... \_\_args)
- template<typename... \_Args>  
    **iterator emplace\_hint** (const\_iterator \_\_hint, \_Args &&... \_\_args)
- **iterator end** () noexcept
- **const\_iterator end** () const noexcept
- **local\_iterator end** (size\_type \_\_b)
- **const\_local\_iterator end** (size\_type \_\_b) const
- **std::pair< iterator, iterator > equal\_range** (const key\_type &\_\_key)
- **std::pair< const\_iterator, const\_iterator > equal\_range** (const key\_type &\_\_key) const
- size\_type **erase** (const key\_type &\_\_key)
- **iterator erase** (const\_iterator \_\_it)
- **iterator erase** (iterator \_\_it)
- **iterator erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- **iterator find** (const key\_type &\_\_key)
- **const\_iterator find** (const key\_type &\_\_key) const
- **iterator insert** (const value\_type &\_\_obj)
- **iterator insert** (const\_iterator \_\_hint, const value\_type &\_\_obj)
- **iterator insert** (value\_type &&\_\_obj)
- **iterator insert** (const\_iterator \_\_hint, value\_type &&\_\_obj)
- void **insert** (std::initializer\_list< value\_type > \_\_l)
- template<typename \_InputIterator >  
    void **insert** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- float **max\_load\_factor** () const noexcept
- void **max\_load\_factor** (float \_\_f)
- **unordered\_multiset & operator=** (const unordered\_multiset &)=default
- **unordered\_multiset & operator=** (unordered\_multiset &&)=default
- **unordered\_multiset & operator=** (initializer\_list< value\_type > \_\_l)
- void **swap** (unordered\_multiset &\_\_x) noexcept(noexcept(declval< \_Base & >().swap(\_\_x)))

#### Public Attributes

- \_Safe\_iterator\_base \* **\_M\_const\_iterators**
- \_Safe\_iterator\_base \* **\_M\_const\_local\_iterators**
- \_Safe\_iterator\_base \* **\_M\_iterators**
- \_Safe\_iterator\_base \* **\_M\_local\_iterators**
- unsigned int **\_M\_version**

#### Protected Member Functions

- void **\_M\_detach\_all** ()
- void **\_M\_detach\_singular** ()
- \_\_gnu\_cxx::\_\_mutex & **\_M\_get\_mutex** () throw ()
- void **\_M\_invalidate\_all** ()
- void **\_M\_invalidate\_all** () const
- void **\_M\_invalidate\_if** (\_Predicate \_\_pred)
- void **\_M\_invalidate\_local\_if** (\_Predicate \_\_pred)
- void **\_M\_invalidate\_locals** ()
- void **\_M\_revalidate\_singular** ()
- \_Safe\_container & **\_M\_safe** () noexcept
- void **\_M\_swap** (\_Safe\_unordered\_container\_base &\_\_x) noexcept
- void **\_M\_swap** (\_Safe\_sequence\_base &\_\_x) noexcept

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
`class ::__gnu_debug::_Safe_iterator`
- `template<typename _ItT, typename _SeqT>`  
`class ::__gnu_debug::_Safe_local_iterator`

### 4.1039.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
```

Class `std::unordered_multiset` with safety/checking/debug instrumentation.

Definition at line 42 of file `debug/unordered_set`.

### 4.1039.2 Member Function Documentation

#### 4.1039.2.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

#### 4.1039.2.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

## Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 4.1039.2.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map<_Key, _Tp, _Compare, _Allocator> >::_M_transfer_from_if()`.

#### 4.1039.2.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

#### 4.1039.2.5 `_M_invalidate_if()`

```
void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc >
>::_M_invalidate_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_unordered_container.tcc`.

#### 4.1039.2.6 `_M_invalidate_local_if()`

```
void __gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, _Hash, _Pred, _Alloc >
>::_M_invalidate_local_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

Definition at line 70 of file `safe_unordered_container.tcc`.

#### 4.1039.2.7 `_M_revalidate_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

#### 4.1039.2.8 `_M_swap()` [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.



#### 4.1039.2.9 \_M\_swap() [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.1039.3 Member Data Documentation

#### 4.1039.3.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.1039.3.2 \_M\_const\_local\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file safe\_unordered\_base.h.

#### 4.1039.3.3 \_M\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file safe\_base.h.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

#### 4.1039.3.4 `_M_local_iterators`

`_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators` [inherited]

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

#### 4.1039.3.5 `_M_version`

`unsigned int __gnu_debug::_Safe_sequence_base::_M_version` [mutable], [inherited]

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

### 4.1040 `std::unordered_set<_Value, _Hash, _Pred, _Alloc>` Class Template Reference

#### Public Types

- `typedef _Hashtable::key_type` [key\\_type](#)
- `typedef _Hashtable::value_type` [value\\_type](#)
- `typedef _Hashtable::hasher` [hasher](#)
- `typedef _Hashtable::key_equal` [key\\_equal](#)
- `typedef _Hashtable::allocator_type` [allocator\\_type](#)
- `typedef _Hashtable::pointer` [pointer](#)
- `typedef _Hashtable::const_pointer` [const\\_pointer](#)
- `typedef _Hashtable::reference` [reference](#)
- `typedef _Hashtable::const_reference` [const\\_reference](#)
- `typedef _Hashtable::iterator` [iterator](#)
- `typedef _Hashtable::const_iterator` [const\\_iterator](#)
- `typedef _Hashtable::local_iterator` [local\\_iterator](#)
- `typedef _Hashtable::const_local_iterator` [const\\_local\\_iterator](#)
- `typedef _Hashtable::size_type` [size\\_type](#)
- `typedef _Hashtable::difference_type` [difference\\_type](#)

## Public Member Functions

- [unordered\\_set](#) ()=default
- [unordered\\_set](#) (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_set](#) (const unordered\_set &)=default
- [unordered\\_set](#) (unordered\_set &&)=default
- [unordered\\_set](#) (const allocator\_type &\_\_a)
- [unordered\\_set](#) (const unordered\_set &\_\_uset, const allocator\_type &\_\_a)
- [unordered\\_set](#) (unordered\_set &&\_\_uset, const allocator\_type &\_\_a)
- [unordered\\_set](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- [unordered\\_set](#) (size\_type \_\_n, const allocator\_type &\_\_a)
- [unordered\\_set](#) (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
[unordered\\_set](#) (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- [unordered\\_set](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- [unordered\\_set](#) (initializer\_list< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- size\_type bucket (const key\_type &\_\_key) const
- size\_type bucket\_count () const noexcept
- size\_type bucket\_size (size\_type \_\_n) const
- const\_iterator cbegin () const noexcept
- const\_iterator cend () const noexcept
- void clear () noexcept
- size\_type count (const key\_type &\_\_x) const
- template<typename... \_Args>  
[std::pair](#)< iterator, bool > [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator](#) [emplace\\_hint](#) (const\_iterator \_\_pos, \_Args &&... \_\_args)
- bool empty () const noexcept
- size\_type erase (const key\_type &\_\_x)
- iterator erase (const\_iterator \_\_first, const\_iterator \_\_last)
- allocator\_type get\_allocator () const noexcept
- hasher hash\_function () const
- template<typename \_InputIterator >  
void [insert](#) (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void [insert](#) (initializer\_list< value\_type > \_\_l)
- key\_equal key\_eq () const
- float load\_factor () const noexcept
- size\_type max\_bucket\_count () const noexcept
- float max\_load\_factor () const noexcept
- void max\_load\_factor (float \_\_z)
- size\_type max\_size () const noexcept
- unordered\_set & operator= (const unordered\_set &)=default

- `unordered_set & operator= (unordered_set &&)=default`
  - `unordered_set & operator= (initializer_list< value_type > __l)`
  - `void rehash (size_type __n)`
  - `void reserve (size_type __n)`
  - `size_type size () const noexcept`
  - `void swap (unordered_set &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))`
- 
- `iterator begin () noexcept`
  - `const_iterator begin () const noexcept`
- 
- `iterator end () noexcept`
  - `const_iterator end () const noexcept`
- 
- `std::pair< iterator, bool > insert (const value_type &__x)`
  - `std::pair< iterator, bool > insert (value_type &&__x)`
- 
- `iterator insert (const_iterator __hint, const value_type &__x)`
  - `iterator insert (const_iterator __hint, value_type &&__x)`
- 
- `iterator erase (const_iterator __position)`
  - `iterator erase (iterator __position)`
- 
- `iterator find (const key_type &__x)`
  - `const_iterator find (const key_type &__x) const`
- 
- `std::pair< iterator, iterator > equal_range (const key_type &__x)`
  - `std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const`
- 
- `local_iterator begin (size_type __n)`
  - `const_local_iterator begin (size_type __n) const`
  - `const_local_iterator cbegin (size_type __n) const`
- 
- `local_iterator end (size_type __n)`
  - `const_local_iterator end (size_type __n) const`
  - `const_local_iterator cend (size_type __n) const`

## Friends

- template<typename \_Value1, typename \_Hash1, typename \_Pred1, typename \_Alloc1 >  
bool **operator==** (const unordered\_set< \_Value1, \_Hash1, \_Pred1, \_Alloc1 > &, const unordered\_set< \_Value1, \_Hash1, \_Pred1, \_Alloc1 > &)

## 4.1040.1 Detailed Description

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc =
allocator<_Value>>
```

```
class std::unordered_set< _Value, _Hash, _Pred, _Alloc >
```

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

## Template Parameters

|               |                                                               |
|---------------|---------------------------------------------------------------|
| <i>_Value</i> | Type of key objects.                                          |
| <i>_Hash</i>  | Hashing function object type, defaults to hash<_Value>.       |
| <i>_Pred</i>  | Predicate function object type, defaults to equal_to<_Value>. |
| <i>_Alloc</i> | Allocator type, defaults to allocator<_Key>.                  |

Meets the requirements of a [container](#), and [unordered associative container](#)

Base is \_Hashtable, dispatched at compile time via template alias \_\_uset\_hashtable.

Definition at line 97 of file unordered\_set.h.

## 4.1040.2 Member Typedef Documentation

## 4.1040.2.1 allocator\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

Definition at line 110 of file unordered\_set.h.

#### 4.1040.2.2 `const_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

Definition at line 120 of file `unordered_set.h`.

#### 4.1040.2.3 `const_local_iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >↵
::const_local_iterator
```

Iterator-related typedefs.

Definition at line 122 of file `unordered_set.h`.

#### 4.1040.2.4 `const_pointer`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_pointer
```

Iterator-related typedefs.

Definition at line 116 of file `unordered_set.h`.

#### 4.1040.2.5 `const_reference`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_reference
```

Iterator-related typedefs.

Definition at line 118 of file `unordered_set.h`.

#### 4.1040.2.6 `difference_type`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_set<_Value, _Hash, _Pred, _Alloc>::difference_type
```

Iterator-related typedefs.

Definition at line 124 of file `unordered_set.h`.

#### 4.1040.2.7 `hasher`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_set<_Value, _Hash, _Pred, _Alloc>::hasher
```

Public typedefs.

Definition at line 108 of file `unordered_set.h`.

#### 4.1040.2.8 `iterator`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc>::iterator
```

Iterator-related typedefs.

Definition at line 119 of file `unordered_set.h`.

#### 4.1040.2.9 `key_equal`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_set<_Value, _Hash, _Pred, _Alloc>::key_equal
```

Public typedefs.

Definition at line 109 of file `unordered_set.h`.

#### 4.1040.2.10 key\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

Definition at line 106 of file unordered\_set.h.

#### 4.1040.2.11 local\_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::local_iterator
```

Iterator-related typedefs.

Definition at line 121 of file unordered\_set.h.

#### 4.1040.2.12 pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

Definition at line 115 of file unordered\_set.h.

#### 4.1040.2.13 reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

Definition at line 117 of file unordered\_set.h.



## 4.1040.2.14 size\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

Definition at line 123 of file unordered\_set.h.

## 4.1040.2.15 value\_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

Definition at line 107 of file unordered\_set.h.

## 4.1040.3 Constructor &amp; Destructor Documentation

## 4.1040.3.1 unordered\_set() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set () [default]
```

Default constructor.

## 4.1040.3.2 unordered\_set() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 size_type __n,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Default constructor creates no elements.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eqf</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Definition at line 145 of file `unordered_set.h`.

**4.1040.3.3 `unordered_set()`** [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 _InputIterator __first,
 _InputIterator __last,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eqf = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an `unordered_set` from a range.

**Parameters**

|                      |                                    |
|----------------------|------------------------------------|
| <code>__first</code> | An input iterator.                 |
| <code>__last</code>  | An input iterator.                 |
| <code>__n</code>     | Minimal initial number of buckets. |
| <code>__hf</code>    | A hash functor.                    |
| <code>__eqf</code>   | A key equality functor.            |
| <code>__a</code>     | An allocator object.               |

Create an `unordered_set` consisting of copies of the elements from `[__first,__last)`. This is linear in `N` (where `N` is `distance(__first,__last)`).

Definition at line 166 of file `unordered_set.h`.

**4.1040.3.4 `unordered_set()`** [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 const unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy constructor.

## 4.1040.3.5 unordered\_set() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move constructor.

## 4.1040.3.6 unordered\_set() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 const allocator_type & __a) [inline], [explicit]
```

Creates an unordered\_set with no elements.

## Parameters

|                  |                      |
|------------------|----------------------|
| $\leftarrow$     | An allocator object. |
| <code>__a</code> |                      |

Definition at line 185 of file unordered\_set.h.

## 4.1040.3.7 unordered\_set() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
 initializer_list< value_type > __l,
 size_type __n = 0,
 const hasher & __hf = hasher(),
 const key_equal & __eq1 = key_equal(),
 const allocator_type & __a = allocator_type()) [inline]
```

Builds an unordered\_set from an initializer\_list.

## Parameters

|                    |                                    |
|--------------------|------------------------------------|
| <code>__l</code>   | An initializer_list.               |
| <code>__n</code>   | Minimal initial number of buckets. |
| <code>__hf</code>  | A hash functor.                    |
| <code>__eq1</code> | A key equality functor.            |
| <code>__a</code>   | An allocator object.               |

Create an `unordered_set` consisting of copies of the elements in the list. This is linear in  $N$  (where  $N$  is `l.size()`).

Definition at line 220 of file `unordered_set.h`.

#### 4.1040.4 Member Function Documentation

##### 4.1040.4.1 `begin()` [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 319 of file `unordered_set.h`.

##### 4.1040.4.2 `begin()` [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the `unordered_set`.

Definition at line 323 of file `unordered_set.h`.

##### 4.1040.4.3 `begin()` [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

#### Parameters

|                  |                   |
|------------------|-------------------|
| <code>__n</code> | The bucket index. |
|------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 737 of file unordered\_set.h.

**4.1040.4.4 begin()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                   |                   |
|-------------------|-------------------|
| $\leftrightarrow$ | The bucket index. |
| $n$               |                   |

**Returns**

A read-only local iterator.

Definition at line 741 of file unordered\_set.h.

**4.1040.4.5 bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count () const [inline],
[noexcept]
```

Returns the number of buckets of the unordered\_set.

Definition at line 703 of file unordered\_set.h.

**4.1040.4.6 cbegin()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered\_set.

Definition at line 346 of file unordered\_set.h.

**4.1040.4.7   cbegin()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

**Parameters**

|                                  |                   |
|----------------------------------|-------------------|
| $\leftarrow$<br><code>__n</code> | The bucket index. |
|----------------------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 745 of file `unordered_set.h`.

**4.1040.4.8   cend()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 354 of file `unordered_set.h`.

**4.1040.4.9   cend()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|                                  |                   |
|----------------------------------|-------------------|
| $\leftarrow$<br><code>__n</code> | The bucket index. |
|----------------------------------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 765 of file unordered\_set.h.

**4.1040.4.10 clear()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear () [inline], [noexcept]
```

Erases all elements in an unordered\_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 572 of file unordered\_set.h.

**4.1040.4.11 count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
 const key_type & __x) const [inline]
```

Finds the number of elements.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__x</code> | Element to located. |
|------------------|---------------------|

**Returns**

Number of elements with specified key.

This function only makes sense for unordered\_multisets; for unordered\_set the result will either be 0 (not present) or 1 (present).

Definition at line 667 of file unordered\_set.h.

#### 4.1040.4.12 `emplace()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
std::pair<iterator, bool> std::unordered_set<_Value, _Hash, _Pred, _Alloc >::emplace (
 _Args &&... __args) [inline]
```

Attempts to build and insert an element into the `unordered_set`.

##### Parameters

|                     |                                        |
|---------------------|----------------------------------------|
| <code>__args</code> | Arguments used to generate an element. |
|---------------------|----------------------------------------|

##### Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to build and insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 376 of file `unordered_set.h`.

#### 4.1040.4.13 `emplace_hint()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::emplace_hint (
 const_iterator __pos,
 _Args &&... __args) [inline]
```

Attempts to insert an element into the `unordered_set`.

##### Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__pos</code>  | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__args</code> | Arguments used to generate the element to be inserted.                        |

##### Returns

An iterator that points to the element with key equivalent to the one generated from `__args` (may or may not be the element itself).



This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `emplace()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant time.

Definition at line 402 of file `unordered_set.h`.

#### 4.1040.4.14 `empty()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [noexcept]
```

Returns true if the `unordered_set` is empty.

Definition at line 298 of file `unordered_set.h`.

#### 4.1040.4.15 `end()` [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 333 of file `unordered_set.h`.

#### 4.1040.4.16 `end()` [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the `unordered_set`.

Definition at line 337 of file `unordered_set.h`.

#### 4.1040.4.17 `end()` [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 757 of file unordered\_set.h.

**4.1040.4.18 end()** [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::end (
 size_type __n) const [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

**Parameters**

|       |                   |
|-------|-------------------|
| $\_n$ | The bucket index. |
|-------|-------------------|

**Returns**

A read-only local iterator.

Definition at line 761 of file unordered\_set.h.

**4.1040.4.19 equal\_range()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, iterator> std::unordered_set<_Value, _Hash, _Pred, _Alloc >::equal_range (
 const key_type & __x) [inline]
```

Finds a subsequence matching given key.

**Parameters**

|       |                    |
|-------|--------------------|
| $\_x$ | Key to be located. |
|-------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 691 of file unordered\_set.h.

**4.1040.4.20 equal\_range()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<const_iterator, const_iterator> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::
equal_range (
 const key_type & __x) const [inline]
```

Finds a subsequence matching given key.

**Parameters**

|                  |                    |
|------------------|--------------------|
| <code>__x</code> | Key to be located. |
|------------------|--------------------|

**Returns**

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

Definition at line 695 of file unordered\_set.h.

**4.1040.4.21 erase()** [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __position) [inline]
```

Erases an element from an unordered\_set.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 522 of file `unordered_set.h`.

**4.1040.4.22 erase()** [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 iterator __position) [inline]
```

Erases an element from an `unordered_set`.

**Parameters**

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__position</code> | An iterator pointing to the element to be erased. |
|-------------------------|---------------------------------------------------|

**Returns**

An iterator pointing to the element immediately following `__position` prior to the element being erased. If no such element exists, `end()` is returned.

This function erases an element, pointed to by the given iterator, from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 527 of file `unordered_set.h`.

**4.1040.4.23 erase()** [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const key_type & __x) [inline]
```

Erases elements according to the provided key.

## Parameters

|                    |                              |
|--------------------|------------------------------|
| <code>__key</code> | Key of element to be erased. |
| <code>__x</code>   |                              |

## Returns

The number of elements erased.

This function erases all the elements located by the given key from an `unordered_set`. For an `unordered_set` the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 544 of file `unordered_set.h`.

4.1040.4.24 `erase()` [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Erases a [`__first`,`__last`) range of elements from an `unordered_set`.

## Parameters

|                      |                                                           |
|----------------------|-----------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be erased. |
| <code>__last</code>  | Iterator pointing to the end of the range to be erased.   |

## Returns

The iterator `__last`.

This function erases a sequence of elements from an `unordered_set`. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 562 of file `unordered_set.h`.

**4.1040.4.25** `find()` [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set<_Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) [inline]
```

Tries to locate an element in an `unordered_set`.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 649 of file `unordered_set.h`.

## 4.1040.4.26 find() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
 const key_type & __x) const [inline]
```

Tries to locate an element in an `unordered_set`.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <code>_↔</code> | Element to be located. |
| <code>_X</code> |                        |

## Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end ( `end()` ) iterator.

Definition at line 653 of file `unordered_set.h`.

## 4.1040.4.27 get\_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator () const [inline],
[noexcept]
```

Returns the allocator object used by the `unordered_set`.

Definition at line 291 of file `unordered_set.h`.

**4.1040.4.28** `hash_function()`

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function () const [inline]
```

Returns the hash functor object with which the `unordered_set` was constructed.

Definition at line 625 of file `unordered_set.h`.

**4.1040.4.29** `insert()` [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const value_type & __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

**Returns**

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a `bool` that is true if the element was actually inserted.

This function attempts to insert an element into the `unordered_set`. An `unordered_set` relies on unique keys and thus an element is only inserted if it is not already present in the `unordered_set`.

Insertion requires amortized constant time.

Definition at line 420 of file `unordered_set.h`.

**4.1040.4.30** `insert()` [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair<iterator, bool> std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 value_type && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.



## Parameters

|                  |                         |
|------------------|-------------------------|
| <code>__x</code> | Element to be inserted. |
|------------------|-------------------------|

## Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered\_set. An unordered\_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered\_set.

Insertion requires amortized constant time.

Definition at line 424 of file unordered\_set.h.

References std::move().

## 4.1040.4.31 insert() [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 const value_type & __x) [inline]
```

Attempts to insert an element into the unordered\_set.

## Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

## Returns

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↵html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 449 of file unordered\_set.h.

**4.1040.4.32** `insert()` [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 const_iterator __hint,
 value_type && __x) [inline]
```

Attempts to insert an element into the `unordered_set`.

**Parameters**

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <code>__hint</code> | An iterator that serves as a hint as to where the element should be inserted. |
| <code>__x</code>    | Element to be inserted.                                                       |

**Returns**

An iterator that points to the element with key of `__x` (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument `insert()` does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.

For more on *hinting*, see: [https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↔html#containers.associative.insert\\_hints](https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↔html#containers.associative.insert_hints)

Insertion requires amortized constant.

Definition at line 453 of file `unordered_set.h`.

References `std::move()`.

**4.1040.4.33** `insert()` [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator >
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

A template function that attempts to insert a range of elements.

**Parameters**

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the start of the range to be inserted. |
| <code>__last</code>  | Iterator pointing to the end of the range.                  |

Complexity similar to that of the range constructor.

Definition at line 468 of file unordered\_set.h.

#### 4.1040.4.34 insert() [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
 initializer_list< value_type > __l) [inline]
```

Attempts to insert a list of elements into the unordered\_set.

##### Parameters

|   |                                                                 |
|---|-----------------------------------------------------------------|
| ↔ | A std::initializer_list<value_type> of elements to be inserted. |
| ↔ |                                                                 |
| ↔ |                                                                 |
| ↔ |                                                                 |
| / |                                                                 |

Complexity similar to that of the range constructor.

Definition at line 479 of file unordered\_set.h.

#### 4.1040.4.35 key\_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq () const [inline]
```

Returns the key comparison object with which the unordered\_set was constructed.

Definition at line 631 of file unordered\_set.h.

#### 4.1040.4.36 load\_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor () const [inline], [noexcept]
```

Returns the average number of elements per bucket.

Definition at line 773 of file unordered\_set.h.

**4.1040.4.37 max\_bucket\_count()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered\_set.

Definition at line 708 of file unordered\_set.h.

**4.1040.4.38 max\_load\_factor()** [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const [inline],
[noexcept]
```

Returns a positive number that the unordered\_set tries to keep the load factor less than or equal to.

Definition at line 779 of file unordered\_set.h.

**4.1040.4.39 max\_load\_factor()** [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
 float __z) [inline]
```

Change the unordered\_set maximum load factor.

**Parameters**

|           |                              |
|-----------|------------------------------|
| <b>_z</b> | The new maximum load factor. |
|-----------|------------------------------|

Definition at line 787 of file unordered\_set.h.

**4.1040.4.40 max\_size()**

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the maximum size of the unordered\_set.

Definition at line 308 of file unordered\_set.h.

#### 4.1040.4.41 operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 const unordered_set< _Value, _Hash, _Pred, _Alloc > &) [default]
```

Copy assignment operator.

#### 4.1040.4.42 operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 unordered_set< _Value, _Hash, _Pred, _Alloc > &&) [default]
```

Move assignment operator.

#### 4.1040.4.43 operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set& std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Unordered\_set list assignment operator.

##### Parameters

|   |                      |
|---|----------------------|
| ↩ | An initializer_list. |
| ↩ |                      |
| ↩ |                      |
| ↩ |                      |
| / |                      |

This function fills an unordered\_set with copies of the elements in the initializer list \_\_l.

Note that the assignment completely changes the unordered\_set and that the resulting unordered\_set's size is the same as the number of elements assigned.

Definition at line 283 of file unordered\_set.h.

#### 4.1040.4.44 rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
 size_type __n) [inline]
```

May rehash the unordered\_set.

##### Parameters

|                  |                            |
|------------------|----------------------------|
| <code>__n</code> | The new number of buckets. |
|------------------|----------------------------|

Rehash will occur only if the new number of buckets respect the unordered\_set maximum load factor.

Definition at line 798 of file unordered\_set.h.

#### 4.1040.4.45 reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
 size_type __n) [inline]
```

Prepare the unordered\_set for a specified number of elements.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

Same as rehash(ceil(n / max\_load\_factor())).

Definition at line 809 of file unordered\_set.h.

#### 4.1040.4.46 size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size () const [inline], [noexcept]
```

Returns the size of the unordered\_set.

Definition at line 303 of file unordered\_set.h.

#### 4.1040.4.47 swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
 unordered_set< _Value, _Hash, _Pred, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another unordered\_set.

#### Parameters

|                  |                                                           |
|------------------|-----------------------------------------------------------|
| <code>__x</code> | An unordered_set of the same element and allocator types. |
|------------------|-----------------------------------------------------------|

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.

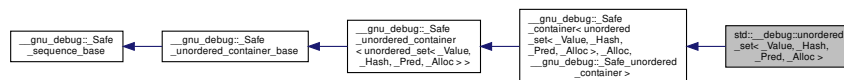
Definition at line 585 of file unordered\_set.h.

The documentation for this class was generated from the following file:

- [unordered\\_set.h](#)

## 4.1041 std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc > Class Template Reference

Inheritance diagram for std::\_\_debug::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >:



#### Public Types

- typedef \_Base::allocator\_type **allocator\_type**
- typedef \_\_gnu\_debug::Safe\_iterator< \_Base\_const\_iterator, unordered\_set > **const\_iterator**
- typedef \_\_gnu\_debug::Safe\_local\_iterator< \_Base\_const\_local\_iterator, unordered\_set > **const\_local\_iterator**
- typedef \_Base::hasher **hasher**
- typedef \_\_gnu\_debug::Safe\_iterator< \_Base\_iterator, unordered\_set > **iterator**
- typedef \_Base::key\_equal **key\_equal**
- typedef \_Base::key\_type **key\_type**
- typedef \_\_gnu\_debug::Safe\_local\_iterator< \_Base\_local\_iterator, unordered\_set > **local\_iterator**
- typedef \_Base::size\_type **size\_type**
- typedef \_Base::value\_type **value\_type**

## Public Member Functions

- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (const [unordered\\_set](#) &)=default
- **unordered\_set** (const [\\_Base](#) &\_\_x)
- **unordered\_set** ([unordered\\_set](#) &&)=default
- **unordered\_set** (const allocator\_type &\_\_a)
- **unordered\_set** (const [unordered\\_set](#) &\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([unordered\\_set](#) &&\_\_uset, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n=0, const hasher &\_\_hf=hasher(), const key\_↵equal &\_\_eq=key\_equal(), const allocator\_type &\_\_a=allocator\_type())
- **unordered\_set** (size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** (size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const allocator\_type &\_\_a)
- template<typename \_InputIterator >  
**unordered\_set** (\_InputIterator \_\_first, \_InputIterator \_\_last, size\_type \_\_n, const hasher &\_\_hf, const allocator\_↵type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const allocator\_type &\_\_a)
- **unordered\_set** ([initializer\\_list](#)< value\_type > \_\_l, size\_type \_\_n, const hasher &\_\_hf, const allocator\_type &↵\_\_a)
- [\\_Base](#) & [\\_M\\_base](#) () noexcept
- const [\\_Base](#) & [\\_M\\_base](#) () const noexcept
- void [\\_M\\_swap](#) (\_Safe\_container &\_\_x) noexcept
- [iterator begin](#) () noexcept
- [const\\_iterator begin](#) () const noexcept
- [local\\_iterator begin](#) (size\_type \_\_b)
- [const\\_local\\_iterator begin](#) (size\_type \_\_b) const
- size\_type [bucket\\_size](#) (size\_type \_\_b) const
- [const\\_iterator cbegin](#) () const noexcept
- [const\\_local\\_iterator cbegin](#) (size\_type \_\_b) const
- [const\\_iterator cend](#) () const noexcept
- [const\\_local\\_iterator cend](#) (size\_type \_\_b) const
- void [clear](#) () noexcept
- template<typename... \_Args>  
[std::pair](#)< [iterator](#), bool > [emplace](#) (\_Args &&... \_\_args)
- template<typename... \_Args>  
[iterator emplace\\_hint](#) ([const\\_iterator](#) \_\_hint, \_Args &&... \_\_args)
- [iterator end](#) () noexcept
- [const\\_iterator end](#) () const noexcept
- [local\\_iterator end](#) (size\_type \_\_b)
- [const\\_local\\_iterator end](#) (size\_type \_\_b) const
- [std::pair](#)< [iterator](#), [iterator](#) > [equal\\_range](#) (const key\_type &\_\_key)
- [std::pair](#)< [const\\_iterator](#), [const\\_iterator](#) > [equal\\_range](#) (const key\_type &\_\_key) const
- size\_type [erase](#) (const key\_type &\_\_key)
- [iterator erase](#) ([const\\_iterator](#) \_\_it)
- [iterator erase](#) ([iterator](#) \_\_it)



- `iterator erase` (`const_iterator` \_\_first, `const_iterator` \_\_last)
- `iterator find` (`const key_type` &\_\_key)
- `const_iterator find` (`const key_type` &\_\_key) `const`
- `std::pair< iterator, bool > insert` (`const value_type` &\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, `const value_type` &\_\_obj)
- `std::pair< iterator, bool > insert` (`value_type` &&\_\_obj)
- `iterator insert` (`const_iterator` \_\_hint, `value_type` &&\_\_obj)
- `void insert` (`std::initializer_list`< `value_type` > \_\_l)
- `template<typename _InputIterator >`  
`void insert` (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- `float max_load_factor` () `const noexcept`
- `void max_load_factor` (`float` \_\_f)
- `unordered_set` & `operator=` (`const unordered_set` &)=default
- `unordered_set` & `operator=` (`unordered_set` &&)=default
- `unordered_set` & `operator=` (`initializer_list`< `value_type` > \_\_l)
- `void swap` (`unordered_set` &\_\_x) `noexcept(noexcept(declval<_Base &>().swap(__x)))`

#### Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_const_local_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- `_Safe_iterator_base` \* `_M_local_iterators`
- `unsigned int` `_M_version`

#### Protected Member Functions

- `void` `_M_detach_all` ()
- `void` `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () `throw ()`
- `void` `_M_invalidate_all` ()
- `void` `_M_invalidate_all` () `const`
- `void` `_M_invalidate_if` (`_Predicate` \_\_pred)
- `void` `_M_invalidate_local_if` (`_Predicate` \_\_pred)
- `void` `_M_invalidate_locals` ()
- `void` `_M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () `noexcept`
- `void` `_M_swap` (`_Safe_unordered_container_base` &\_\_x) `noexcept`
- `void` `_M_swap` (`_Safe_sequence_base` &\_\_x) `noexcept`

#### Friends

- `template<typename _ItT, typename _SeqT, typename _CatT >`  
`class` `::__gnu_debug::Safe_iterator`
- `template<typename _ItT, typename _SeqT >`  
`class` `::__gnu_debug::Safe_local_iterator`

#### 4.1041.1 Detailed Description

```
template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc =
std::allocator<_Value>>>
class std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc >
```

Class std::unordered\_set with safety/checking/debug instrumentation.

Definition at line 40 of file debug/unordered\_set.

#### 4.1041.2 Member Function Documentation

##### 4.1041.2.1 \_M\_detach\_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

##### 4.1041.2.2 \_M\_detach\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators i attached to this sequence, i->\_M\_version == \_M\_version.

##### 4.1041.2.3 \_M\_get\_mutex()

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in \_Safe\_sequence.

Referenced by \_\_gnu\_debug::\_Safe\_sequence< map< \_Key, \_Tp, \_Compare, \_Allocator > >::\_M\_transfer\_from\_if().

## 4.1041.2.4 \_M\_invalidate\_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file safe\_base.h.

References `__gnu_debug::_Safe_sequence_base::_M_version`.

## 4.1041.2.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::↵
_M_invalidate_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file safe\_unordered\_container.tcc.

## 4.1041.2.6 \_M\_invalidate\_local\_if()

```
void __gnu_debug::_Safe_unordered_container< unordered_set< _Value, _Hash, _Pred, _Alloc > >::↵
_M_invalidate_local_if (
 _Predicate __pred) [protected], [inherited]
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal local iterators nested in the safe ones.

Definition at line 70 of file safe\_unordered\_container.tcc.

## 4.1041.2.7 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.1041.2.8 \_M\_swap() [1/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
 _Safe_unordered_container_base & __x) [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

#### 4.1041.2.9 `_M_swap()` [2/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

### 4.1041.3 Member Data Documentation

#### 4.1041.3.1 `_M_const_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1041.3.2 `_M_const_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators [inherited]
```

The list of constant local iterators that reference this container.

Definition at line 130 of file `safe_unordered_base.h`.

#### 4.1041.3.3 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

4.1041.3.4 `_M_local_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators [inherited]
```

The list of mutable local iterators that reference this container.

Definition at line 127 of file `safe_unordered_base.h`.

4.1041.3.5 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/unordered\\_set](#)

4.1042 `std::uses_allocator<_Tp, _Alloc>` Struct Template Reference

Inherits type `<_Tp, _Alloc>`.

## 4.1042.1 Detailed Description

```
template<typename _Tp, typename _Alloc>
struct std::uses_allocator<_Tp, _Alloc>
```

Declare `uses_allocator` so it can be specialized in `<queue>` etc.

[allocator.uses.trait]

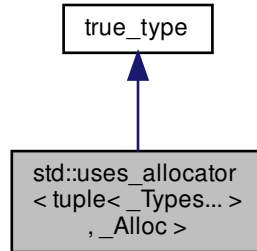
Definition at line 74 of file `memoryfwd.h`.

The documentation for this struct was generated from the following file:

- [memoryfwd.h](#)

#### 4.1043 std::uses\_allocator< tuple< \_Types... >, \_Alloc > Struct Template Reference

Inheritance diagram for std::uses\_allocator< tuple< \_Types... >, \_Alloc >:



##### Public Types

- typedef [integral\\_constant](#)< \_Tp, \_\_v > **type**
- typedef \_Tp **value\_type**

##### Public Member Functions

- constexpr **operator value\_type** () const noexcept
- constexpr value\_type **operator()** () const noexcept

##### Static Public Attributes

- static constexpr \_Tp **value**

##### 4.1043.1 Detailed Description

```
template<typename... _Types, typename _Alloc>
struct std::uses_allocator< tuple< _Types... >, _Alloc >
```

Partial specialization for tuples.

Definition at line 1662 of file tuple.

The documentation for this struct was generated from the following file:

- [tuple](#)

## 4.1044 std::valarray&lt; \_Tp &gt; Class Template Reference

## Public Types

- typedef \_Tp **value\_type**

## Public Member Functions

- [valarray](#) ()
- [valarray](#) (size\_t)
- [valarray](#) (const \_Tp &, size\_t)
- [valarray](#) (const \_Tp \* \_\_restrict\_\_, size\_t)
- [valarray](#) (const [valarray](#) &)
- [valarray](#) ([valarray](#) &&) noexcept
- [valarray](#) (const [slice\\_array](#)< \_Tp > &)
- [valarray](#) (const [gslice\\_array](#)< \_Tp > &)
- [valarray](#) (const [mask\\_array](#)< \_Tp > &)
- [valarray](#) (const [indirect\\_array](#)< \_Tp > &)
- [valarray](#) ([initializer\\_list](#)< \_Tp >)
- template<class \_Dom >  
**valarray** (const \_Expr< \_Dom, \_Tp > &\_\_e)
- template<typename \_Tp>  
**valarray** (const \_Tp \* \_\_restrict\_\_ \_\_p, size\_t \_\_n)
- \_Expr< \_ValFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(\_Tp)) const
- \_Expr< \_RefFunClos< \_ValArray, \_Tp >, \_Tp > [apply](#) (\_Tp func(const \_Tp &)) const
- [valarray](#)< \_Tp > [cshift](#) (int \_\_n) const
- \_Tp [max](#) () const
- \_Tp [min](#) () const
- \_UnaryOp< \_\_logical\_not >::\_Rt [operator!](#) () const
- [valarray](#)< \_Tp > & [operator%=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator%=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator%=>](#) (const \_Expr< \_Dom, \_Tp > &)
- [valarray](#)< \_Tp > & [operator&=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator&=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator&=>](#) (const \_Expr< \_Dom, \_Tp > &)
- [valarray](#)< \_Tp > & [operator\\*=>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator\\*=>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator\\*=>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_unary\_plus >::\_Rt [operator+>](#) () const
- [valarray](#)< \_Tp > & [operator+>](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator+>](#) (const [valarray](#)< \_Tp > &)
- template<class \_Dom >  
[valarray](#)< \_Tp > & [operator+>](#) (const \_Expr< \_Dom, \_Tp > &)
- \_UnaryOp< \_\_negate >::\_Rt [operator->](#) () const
- [valarray](#)< \_Tp > & [operator->](#) (const \_Tp &)
- [valarray](#)< \_Tp > & [operator->](#) (const [valarray](#)< \_Tp > &)

- `template<class _Dom >`  
`valarray<_Tp> & operator-= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator/= (const _Tp &)`
- `valarray<_Tp> & operator/= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator/= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator<=<= (const _Tp &)`
- `valarray<_Tp> & operator<=<= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator<<= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator= (const valarray<_Tp> &__v)`
- `valarray<_Tp> & operator= (valarray<_Tp> &&__v) noexcept`
- `valarray<_Tp> & operator= (const _Tp &__t)`
- `valarray<_Tp> & operator= (const slice_array<_Tp> &__sa)`
- `valarray<_Tp> & operator= (const gslice_array<_Tp> &__ga)`
- `valarray<_Tp> & operator= (const mask_array<_Tp> &__ma)`
- `valarray<_Tp> & operator= (const indirect_array<_Tp> &__ia)`
- `valarray & operator= (initializer_list<_Tp> __l)`
- `template<class _Dom >`  
`valarray<_Tp> & operator= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator>>= (const _Tp &)`
- `valarray<_Tp> & operator>>= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator>>= (const _Expr<_Dom, _Tp> &)`
- `_Tp & operator[] (size_t __i)`
- `const _Tp & operator[] (size_t) const`
- `_Expr<_SClos<_ValArray, _Tp>, _Tp> operator[] (slice __s) const`
- `slice_array<_Tp> operator[] (slice __s)`
- `_Expr<_GClos<_ValArray, _Tp>, _Tp> operator[] (const gslice &__s) const`
- `gslice_array<_Tp> operator[] (const gslice &__s)`
- `valarray<_Tp> operator[] (const valarray<bool> &__m) const`
- `mask_array<_Tp> operator[] (const valarray<bool> &__m)`
- `_Expr<_IClos<_ValArray, _Tp>, _Tp> operator[] (const valarray<size_t> &__i) const`
- `indirect_array<_Tp> operator[] (const valarray<size_t> &__i)`
- `valarray<_Tp> & operator^= (const _Tp &)`
- `valarray<_Tp> & operator^= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator^= (const _Expr<_Dom, _Tp> &)`
- `valarray<_Tp> & operator|= (const _Tp &)`
- `valarray<_Tp> & operator|= (const valarray<_Tp> &)`
- `template<class _Dom >`  
`valarray<_Tp> & operator|= (const _Expr<_Dom, _Tp> &)`
- `_UnaryOp<__bitwise_not>::Rt operator~ () const`
- `void resize (size_t __size, _Tp __c=_Tp())`
- `valarray<_Tp> shift (int __n) const`
- `size_t size () const`
- `_Tp sum () const`
- `void swap (valarray<_Tp> &__v) noexcept`

## Friends

- `class _Array<_Tp>`



## 4.1044.1 Detailed Description

```
template<class _Tp>
class std::valarray< _Tp >
```

Smart array designed to support numeric processing.

A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

## Template Parameters

|                  |                              |
|------------------|------------------------------|
| <code>_Tp</code> | Type of object in the array. |
|------------------|------------------------------|

Definition at line 89 of file valarray.

## 4.1044.2 Constructor &amp; Destructor Documentation

## 4.1044.2.1 valarray()

```
template<class _Tp>
std::valarray< _Tp >::valarray (
 const _Tp * __restrict__,
 size_t)
```

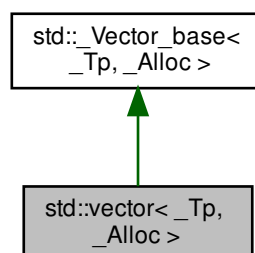
Construct an array initialized to the first  $n$  elements of  $t$ .

The documentation for this class was generated from the following file:

- [valarray](#)

## 4.1045 std::vector&lt; \_Tp, \_Alloc &gt; Class Template Reference

Inheritance diagram for std::vector< \_Tp, \_Alloc >:



## Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `__gnu_cxx::__normal_iterator< const_pointer, vector >` **const\_iterator**
- typedef `_Alloc_traits::const_pointer` **const\_pointer**
- typedef `_Alloc_traits::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `__gnu_cxx::__normal_iterator< pointer, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Alloc_traits::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- [vector](#) ()=default
- [vector](#) (const allocator\_type &\_\_a) noexcept
- [vector](#) (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) (size\_type \_\_n, const value\_type &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- [vector](#) (const [vector](#) &\_\_x)
- [vector](#) ([vector](#) &&) noexcept=default
- [vector](#) (const [vector](#) &\_\_x, const allocator\_type &\_\_a)
- [vector](#) ([vector](#) && \_\_rv, const allocator\_type &\_\_m) noexcept(noexcept([vector](#)(std::declval< [vector](#) && >(), std::declval< const allocator\_type & >(), std::declval< typename `_Alloc_traits::is_always_equal` >())))
- [vector](#) (initializer\_list< value\_type > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename `_InputIterator` , typename = std::\_RequireInputIter< `_InputIterator` >>  
[vector](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last, const allocator\_type &\_\_a=allocator\_type())
- [~vector](#) () noexcept
- template<typename... `_Args`>  
auto **`_M_emplace_aux`** (const\_iterator \_\_position, `_Args` &&... \_\_args) -> iterator
- void [assign](#) (size\_type \_\_n, const value\_type &\_\_val)
- template<typename `_InputIterator` , typename = std::\_RequireInputIter< `_InputIterator` >>  
void [assign](#) (`_InputIterator` \_\_first, `_InputIterator` \_\_last)
- void [assign](#) (initializer\_list< value\_type > \_\_l)
- reference [at](#) (size\_type \_\_n)
- const\_reference [at](#) (size\_type \_\_n) const
- reference [back](#) () noexcept
- const\_reference [back](#) () const noexcept
- iterator [begin](#) () noexcept
- const\_iterator [begin](#) () const noexcept
- size\_type [capacity](#) () const noexcept
- const\_iterator [cbegin](#) () const noexcept
- const\_iterator [cend](#) () const noexcept
- void [clear](#) () noexcept
- const\_reverse\_iterator [crbegin](#) () const noexcept
- const\_reverse\_iterator [crend](#) () const noexcept
- `_Tp` \* [data](#) () noexcept
- const `_Tp` \* **`data`** () const noexcept

- template<typename... \_Args>  
iterator **emplace** (const\_iterator \_\_position, \_Args &&... \_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&... \_\_args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- reference **front** () noexcept
- const\_reference **front** () const noexcept
- allocator\_type **get\_allocator** () const noexcept
- iterator **insert** (const\_iterator \_\_position, const value\_type &\_\_x)
- iterator **insert** (const\_iterator \_\_position, value\_type &&\_\_x)
- iterator **insert** (const\_iterator \_\_position, initializer\_list< value\_type > \_\_l)
- iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const value\_type &\_\_x)
- template<typename \_InputIterator, typename = std::\_RequireInputIter<\_InputIterator>>>  
iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- size\_type **max\_size** () const noexcept
- vector & **operator=** (const vector &\_\_x)
- vector & **operator=** (vector &&\_\_x) noexcept(\_Alloc\_traits::\_S\_nothrow\_move())
- vector & **operator=** (initializer\_list< value\_type > \_\_l)
- reference **operator[]** (size\_type \_\_n) noexcept
- const\_reference **operator[]** (size\_type \_\_n) const noexcept
- void **pop\_back** () noexcept
- void **push\_back** (const value\_type &\_\_x)
- void **push\_back** (value\_type &&\_\_x)
- reverse\_iterator **rbegin** () noexcept
- const\_reverse\_iterator **rbegin** () const noexcept
- reverse\_iterator **rend** () noexcept
- const\_reverse\_iterator **rend** () const noexcept
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size)
- void **resize** (size\_type \_\_new\_size, const value\_type &\_\_x)
- void **shrink\_to\_fit** ()
- size\_type **size** () const noexcept
- void **swap** (vector &\_\_x) noexcept

#### Protected Member Functions

- pointer **\_M\_allocate** (size\_t \_\_n)
- pointer **\_M\_allocate** (size\_t \_\_n)
- template<typename \_ForwardIterator>  
pointer **\_M\_allocate\_and\_copy** (size\_type \_\_n, \_ForwardIterator \_\_first, \_ForwardIterator \_\_last)
- template<typename \_InputIterator>  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, std::input\_iterator\_tag)
- template<typename \_ForwardIterator>  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, std::forward\_iterator\_tag)
- template<typename \_Integer>  
void **\_M\_assign\_dispatch** (\_Integer \_\_n, \_Integer \_\_val, \_\_true\_type)

- `template<typename _InputIterator >`  
`void _M_assign_dispatch (_InputIterator __first, _InputIterator __last, __false_type)`
- `size_type _M_check_len (size_type __n, const char *__s) const`
- `void _M_create_storage (size_t __n)`
- `void _M_deallocate (pointer __p, size_t __n)`
- `void _M_deallocate (pointer __p, size_t __n)`
- `void _M_default_append (size_type __n)`
- `void _M_default_initialize (size_type __n)`
- `template<typename... _Args>`  
`iterator _M_emplace_aux (const_iterator __position, _Args &&... __args)`
- `iterator _M_emplace_aux (const_iterator __position, value_type &&__v)`
- `iterator _M_erase (iterator __position)`
- `iterator _M_erase (iterator __first, iterator __last)`
- `void _M_erase_at_end (pointer __pos) noexcept`
- `void _M_fill_assign (size_type __n, const value_type &__val)`
- `void _M_fill_initialize (size_type __n, const value_type &__value)`
- `void _M_fill_insert (iterator __pos, size_type __n, const value_type &__x)`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `const _Tp_alloc_type & _M_get_Tp_allocator () const noexcept`
- `_Tp_alloc_type & _M_get_Tp_allocator () noexcept`
- `template<typename _Arg >`  
`void _M_insert_aux (iterator __position, _Arg &&__arg)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __val, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `iterator _M_insert_rval (const_iterator __position, value_type &&__v)`
- `void _M_range_check (size_type __n) const`
- `template<typename _InputIterator >`  
`void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator >`  
`void _M_range_insert (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_range_insert (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename... _Args>`  
`void _M_realloc_insert (iterator __position, _Args &&... __args)`
- `bool _M_shrink_to_fit ()`
- `allocator_type get_allocator () const noexcept`

#### Static Protected Member Functions

- `static size_type _S_check_init_len (size_type __n, const allocator_type &__a)`
- `static size_type _S_max_size (const _Tp_alloc_type &__a) noexcept`

#### Protected Attributes

- `_Vector_impl _M_impl`

## 4.1045.1 Detailed Description

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector<_Tp, _Alloc>
```

A standard container which offers fixed time access to individual elements in any order.

## Template Parameters

|                     |                                                                 |
|---------------------|-----------------------------------------------------------------|
| <code>_Tp</code>    | Type of element.                                                |
| <code>_Alloc</code> | Allocator type, defaults to <code>allocator&lt;_Tp&gt;</code> . |

Meets the requirements of a [container](#), a [reversible container](#), and a [sequence](#), including the [optional sequence requirements](#) with the exception of `push_front` and `pop_front`.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

Definition at line 389 of file `std_vector.h`.

## 4.1045.2 Constructor &amp; Destructor Documentation

4.1045.2.1 `vector()` [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector () [default]
```

Creates a vector with no elements.

4.1045.2.2 `vector()` [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::vector (
 const allocator_type & __a) [inline], [explicit], [noexcept]
```

Creates a vector with no elements.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__a</code> | An allocator object. |
|------------------|----------------------|

Definition at line 497 of file `stl_vector.h`.

#### 4.1045.2.3 `vector()` [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 size_type __n,
 const allocator_type & __a = allocator_type()) [inline], [explicit]
```

Creates a vector with default constructed elements.

##### Parameters

|                  |                                             |
|------------------|---------------------------------------------|
| <code>__n</code> | The number of elements to initially create. |
| <code>__a</code> | An allocator.                               |

This constructor fills the vector with `__n` default constructed elements.

Definition at line 510 of file `stl_vector.h`.

#### 4.1045.2.4 `vector()` [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 size_type __n,
 const value_type & __value,
 const allocator_type & __a = allocator_type()) [inline]
```

Creates a vector with copies of an exemplar element.

##### Parameters

|                      |                                             |
|----------------------|---------------------------------------------|
| <code>__n</code>     | The number of elements to initially create. |
| <code>__value</code> | An element to copy.                         |
| <code>__a</code>     | An allocator.                               |

This constructor fills the vector with `__n` copies of `__value`.

Definition at line 522 of file `stl_vector.h`.

## 4.1045.2.5 vector() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector (
 const vector<_Tp, _Alloc > & __x) [inline]
```

Vector copy constructor.

## Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied (i.e. `capacity() == size()` in the new vector).

The newly-created vector uses a copy of the allocator object used by `__x` (unless the allocator traits dictate a different object).

Definition at line 553 of file `stl_vector.h`.

## 4.1045.2.6 vector() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector (
 vector<_Tp, _Alloc > &&) [default], [noexcept]
```

Vector move constructor.

The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

## 4.1045.2.7 vector() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc >::vector (
 const vector<_Tp, _Alloc > & __x,
 const allocator_type & __a) [inline]
```

Copy constructor with alternative allocator.

Definition at line 575 of file `stl_vector.h`.

**4.1045.2.8 vector()** [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 vector< _Tp, _Alloc > && __rv,
 const allocator_type & __m) [inline], [noexcept]
```

Move constructor with alternative allocator.

Definition at line 607 of file `stl_vector.h`.

**4.1045.2.9 vector()** [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
 initializer_list< value_type > __l,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a vector from an initializer list.

**Parameters**

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
| <code>__a</code> | An allocator.        |

Create a vector consisting of copies of the elements in the initializer\_list `__l`.

This will call the element type's copy constructor N times (where N is `__l.size()`) and do no memory reallocation.

Definition at line 625 of file `stl_vector.h`.

**4.1045.2.10 vector()** [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::RequireInputIter<_InputIterator>>
std::vector< _Tp, _Alloc >::vector (
 _InputIterator __first,
 _InputIterator __last,
 const allocator_type & __a = allocator_type()) [inline]
```

Builds a vector from a range.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |
| <code>__a</code>     | An allocator.      |



Create a vector consisting of copies of the elements from `[first,last)`.

If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor  $N$  times (where  $N$  is `distance(first,last)`) and do no memory reallocation. But if only input iterators are used, then this will do at most  $2N$  calls to the copy constructor, and  $\log N$  memory reallocations.

Definition at line 653 of file `std_vector.h`.

#### 4.1045.2.11 `~vector()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector<_Tp, _Alloc>::~~vector () [inline], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 678 of file `std_vector.h`.

### 4.1045.3 Member Function Documentation

#### 4.1045.3.1 `_M_allocate_and_copy()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator >
pointer std::vector<_Tp, _Alloc>::_M_allocate_and_copy (
 size_type __n,
 _ForwardIterator __first,
 _ForwardIterator __last) [inline], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain  $n$  bytes of memory, and then copies `[first,last)` into it.

Definition at line 1508 of file `std_vector.h`.

#### 4.1045.3.2 `_M_range_check()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::_M_range_check (
 size_type __n) const [inline], [protected]
```

Safety check used only from `at()`.

Definition at line 1070 of file `std_vector.h`.

Referenced by `std::vector< sub_match<_Bi_iter>, allocator< sub_match<_Bi_iter>>>::at()`.

#### 4.1045.3.3 `assign()` [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::assign (
 size_type __n,
 const value_type & __val) [inline]
```

Assigns a given value to a vector.

**Parameters**

|                    |                                    |
|--------------------|------------------------------------|
| <code>__n</code>   | Number of elements to be assigned. |
| <code>__val</code> | Value to be assigned.              |

This function fills a vector with `__n` copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 749 of file `stl_vector.h`.

**4.1045.3.4 assign()** [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector<_Tp, _Alloc>::assign (
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Assigns a range to a vector.

**Parameters**

|                      |                    |
|----------------------|--------------------|
| <code>__first</code> | An input iterator. |
| <code>__last</code>  | An input iterator. |

This function fills a vector with copies of the elements in the range `[__first,__last)`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 768 of file `stl_vector.h`.

**4.1045.3.5 assign()** [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::assign (
 initializer_list<value_type> __l) [inline]
```

Assigns an initializer list to a vector.

## Parameters

|                |                      |
|----------------|----------------------|
| <code>↵</code> | An initializer_list. |
| <code>↵</code> |                      |
| <code>↵</code> |                      |
| <code>↵</code> |                      |
| <code>/</code> |                      |

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

Definition at line 794 of file `stl_vector.h`.

4.1045.3.6 `at()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::at (
 size_type __n) [inline]
```

Provides access to the data contained in the vector.

## Parameters

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| <code>↵</code>   | The index of the element for which data should be accessed. |
| <code>__n</code> |                                                             |

## Returns

Read/write reference to data.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 1092 of file `stl_vector.h`.

4.1045.3.7 `at()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::at (
 size_type __n) const [inline]
```

Provides access to the data contained in the vector.

## Parameters

|                                    |                                                             |
|------------------------------------|-------------------------------------------------------------|
| <code>_↵</code><br><code>_n</code> | The index of the element for which data should be accessed. |
|------------------------------------|-------------------------------------------------------------|

## Returns

Read-only (constant) reference to data.

## Exceptions

|                                |                                          |
|--------------------------------|------------------------------------------|
| <code>std::out_of_range</code> | If <code>__n</code> is an invalid index. |
|--------------------------------|------------------------------------------|

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws `out_of_range` if the check fails.

Definition at line 1110 of file `stl_vector.h`.

4.1045.3.8 `back()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector<_Tp, _Alloc>::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

Definition at line 1143 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution<_RealType>::max()`, and `std::piecewise_linear_distribution<↵_RealType>::max()`.

4.1045.3.9 `back()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::back () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

Definition at line 1154 of file `stl_vector.h`.

**4.1045.3.10 begin()** [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
iterator std::vector< _Tp, _Alloc >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 811 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::crend()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::erase()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`, `std::operator<()`, `std::operator==()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rend()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector()`.

**4.1045.3.11 begin()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
const_iterator std::vector< _Tp, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 820 of file `stl_vector.h`.

**4.1045.3.12 capacity()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
size_type std::vector< _Tp, _Alloc >::capacity () const [inline], [noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Definition at line 998 of file `stl_vector.h`.

**4.1045.3.13 cbegin()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
const_iterator std::vector< _Tp, _Alloc >::cbegin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Definition at line 884 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::erase()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::insert()`.

4.1045.3.14 `cend()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cend () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 893 of file `stl_vector.h`.

4.1045.3.15 `clear()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear () [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1498 of file `stl_vector.h`.

4.1045.3.16 `crbegin()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 902 of file `stl_vector.h`.

4.1045.3.17 `crend()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 911 of file `stl_vector.h`.

#### 4.1045.3.18 data()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp* std::vector<_Tp, _Alloc>::data () [inline], [noexcept]
```

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

Definition at line 1168 of file stl\_vector.h.

Referenced by std::regex\_traits<\_CharType>::transform\_primary().

#### 4.1045.3.19 emplace()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
iterator std::vector<_Tp, _Alloc>::emplace (
 const_iterator __position,
 _Args &&... __args) [inline]
```

Inserts an object in vector before specified iterator.

##### Parameters

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__args</code>     | Arguments.                        |

##### Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

Definition at line 1248 of file stl\_vector.h.

#### 4.1045.3.20 empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector<_Tp, _Alloc>::empty () const [inline], [noexcept]
```

Returns true if the vector is empty. (Thus begin() would equal end().)

Definition at line 1007 of file stl\_vector.h.

Referenced by std::piecewise\_constant\_distribution<\_RealType>::densities(), std::piecewise\_linear\_distribution<\_RealType>::densities(), std::piecewise\_constant\_distribution<\_RealType>::intervals(), std::piecewise\_linear\_distribution<\_RealType>::intervals(), std::discrete\_distribution<\_IntType>::max(), std::piecewise\_constant\_distribution<\_RealType>::max(), std::piecewise\_linear\_distribution<\_RealType>::max(), std::piecewise\_constant\_distribution<\_RealType>::min(), std::piecewise\_linear\_distribution<\_RealType>::min(), and std::discrete\_distribution<\_IntType>::probabilities().



**4.1045.3.21** `end()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 829 of file `stl_vector.h`.

Referenced by `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::cbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::empty()`, `std::operator<()`, `std::operator==(, std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::rbegin()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::resize()`, and `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _Bi_iter > > >::vector()`.

**4.1045.3.22** `end()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector<_Tp, _Alloc>::end () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Definition at line 838 of file `stl_vector.h`.

**4.1045.3.23** `erase()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector<_Tp, _Alloc>::erase (
 const_iterator __position) [inline]
```

Remove element at given position.

**Parameters**

|                         |                                            |
|-------------------------|--------------------------------------------|
| <code>__position</code> | Iterator pointing to element to be erased. |
|-------------------------|--------------------------------------------|

**Returns**

An iterator pointing to the next element (or `end()`).

This function will erase the element at the given position and thus shorten the vector by one.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1430 of file `stl_vector.h`.

#### 4.1045.3.24 `erase()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
 const_iterator __first,
 const_iterator __last) [inline]
```

Remove a range of elements.

##### Parameters

|                      |                                                              |
|----------------------|--------------------------------------------------------------|
| <code>__first</code> | Iterator pointing to the first element to be erased.         |
| <code>__last</code>  | Iterator pointing to one past the last element to be erased. |

##### Returns

An iterator pointing to the element pointed to by `__last` prior to erasing (or `end()`).

This function will erase the elements in the range `[__first,__last)` and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using `std::list`. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Definition at line 1457 of file `stl_vector.h`.

#### 4.1045.3.25 `front()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front () [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

Definition at line 1121 of file `stl_vector.h`.

Referenced by `std::piecewise_constant_distribution< _RealType >::min()`, and `std::piecewise_linear_distribution< _RealType >::min()`.

**4.1045.3.26** `front()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector<_Tp, _Alloc>::front () const [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

Definition at line 1132 of file `std_vector.h`.

**4.1045.3.27** `get_allocator()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::_Vector_base<_Tp, _Alloc>::get_allocator [inline], [noexcept]
```

Get a copy of the memory allocation object.

Definition at line 284 of file `std_vector.h`.

**4.1045.3.28** `insert()` [1/5]

```
template<typename _Tp , typename _Alloc >
vector<_Tp, _Alloc>::iterator vector::insert (
 const_iterator __position,
 const value_type & __x)
```

Inserts given value into vector before specified iterator.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__position</code> | A <code>const_iterator</code> into the vector. |
| <code>__x</code>        | Data to be inserted.                           |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 132 of file `vector.tcc`.

**4.1045.3.29** `insert()` [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 value_type && __x) [inline]
```

Inserts given rvalue into vector before specified iterator.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__x</code>        | Data to be inserted.              |

**Returns**

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1293 of file `stl_vector.h`.

**4.1045.3.30** `insert()` [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 initializer_list< value_type > __l) [inline]
```

Inserts an `initializer_list` into the vector.

**Parameters**

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | An iterator into the vector.       |
| <code>__l</code>        | An <code>initializer_list</code> . |

This function will insert copies of the data in the `initializer_list l` into the vector before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1310 of file `stl_vector.h`.

## 4.1045.3.31 insert() [ 4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 size_type __n,
 const value_type & __x) [inline]
```

Inserts a number of copies of given data into the vector.

## Parameters

|                         |                                    |
|-------------------------|------------------------------------|
| <code>__position</code> | A const_iterator into the vector.  |
| <code>__n</code>        | Number of elements to be inserted. |
| <code>__x</code>        | Data to be inserted.               |

## Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1335 of file `stl_vector.h`.

## 4.1045.3.32 insert() [ 5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator , typename = std::_RequireInputIter<_InputIterator>>
iterator std::vector< _Tp, _Alloc >::insert (
 const_iterator __position,
 _InputIterator __first,
 _InputIterator __last) [inline]
```

Inserts a range into the vector.

## Parameters

|                         |                                   |
|-------------------------|-----------------------------------|
| <code>__position</code> | A const_iterator into the vector. |
| <code>__first</code>    | An input iterator.                |
| <code>__last</code>     | An input iterator.                |

**Returns**

An iterator that points to the inserted data.

This function will insert copies of the data in the range `[__first,__last)` into the vector before the location specified by *pos*.

Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using `std::list`.

Definition at line 1379 of file `stl_vector.h`.

**4.1045.3.33 `max_size()`**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const [inline], [noexcept]
```

Returns the `size()` of the largest possible vector.

Definition at line 923 of file `stl_vector.h`.

**4.1045.3.34 `operator=()` [1/3]**

```
template<typename _Tp , typename _Alloc >
vector< _Tp, _Alloc > & vector::operator= (
 const vector< _Tp, _Alloc > & __x)
```

Vector assignment operator.

**Parameters**

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

All the elements of `__x` are copied, but any unused capacity in `__x` will not be copied.

Whether the allocator is copied depends on the allocator traits.

Definition at line 199 of file `vector.tcc`.

**4.1045.3.35 `operator=()` [2/3]**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector& std::vector< _Tp, _Alloc >::operator= (
 vector< _Tp, _Alloc > && __x) [inline], [noexcept]
```

Vector move assignment operator.

## Parameters

|                  |                                                    |
|------------------|----------------------------------------------------|
| <code>__x</code> | A vector of identical element and allocator types. |
|------------------|----------------------------------------------------|

The contents of `__x` are moved into this vector (without copying, if the allocators permit it). Afterwards `__x` is a valid, but unspecified vector.

Whether the allocator is moved depends on the allocator traits.

Definition at line 709 of file `std_vector.h`.

## 4.1045.3.36 operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector& std::vector< _Tp, _Alloc >::operator= (
 initializer_list< value_type > __l) [inline]
```

Vector list assignment operator.

## Parameters

|                  |                      |
|------------------|----------------------|
| <code>__l</code> | An initializer_list. |
|------------------|----------------------|

This function fills a vector with copies of the elements in the initializer list `__l`.

Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.


Definition at line 730 of file `std_vector.h`.

## 4.1045.3.37 operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
 size_type __n) [inline], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                                                                      |                                                             |
|------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| <br><code>_n</code> | The index of the element for which data should be accessed. |
|------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|

**Returns**

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)


Definition at line 1043 of file `stl_vector.h`.

**4.1045.3.38 operator[]()** [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (
 size_type __n) const [inline], [noexcept]
```

Subscript access to the data contained in the vector.

**Parameters**

|                                                                                                        |                                                             |
|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|
| <br><code>_n</code> | The index of the element for which data should be accessed. |
|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|

**Returns**

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and `out_of_range` lookups are not defined. (For checked lookups see `at()`.)

Definition at line 1061 of file `stl_vector.h`.

**4.1045.3.39 pop\_back()**

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back () [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the vector by one.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before `pop_back()` is called.

Definition at line 1225 of file `stl_vector.h`.



4.1045.3.40 `push_back()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector<_Tp, _Alloc>::push_back (
 const value_type & __x) [inline]
```

Add data to the end of the vector.

## Parameters

|                  |                   |
|------------------|-------------------|
| <code>__x</code> | Data to be added. |
|------------------|-------------------|

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

Definition at line 1187 of file `stl_vector.h`.

4.1045.3.41 `rbegin()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rbegin () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 847 of file `stl_vector.h`.

4.1045.3.42 `rbegin()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector<_Tp, _Alloc>::rbegin () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

Definition at line 856 of file `stl_vector.h`.

4.1045.3.43 `rend()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector<_Tp, _Alloc>::rend () [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 865 of file `stl_vector.h`.

**4.1045.3.44** `rend()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

Definition at line 874 of file `stl_vector.h`.

**4.1045.3.45** `reserve()`

```
template<typename _Tp , typename _Alloc >
void vector::reserve (
 size_type __n)
```

Attempt to preallocate enough memory for specified number of elements.

**Parameters**

|                  |                              |
|------------------|------------------------------|
| <code>__n</code> | Number of elements required. |
|------------------|------------------------------|

**Exceptions**

|                                |                                                     |
|--------------------------------|-----------------------------------------------------|
| <code>std::length_error</code> | If <code>n</code> exceeds <code>max_size()</code> . |
|--------------------------------|-----------------------------------------------------|

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than `max_size()`, `length_error` is thrown.

The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

Definition at line 67 of file `vector.tcc`.

**4.1045.3.46** `resize()` [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
 size_type __new_size) [inline]
```

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain. |
|-------------------------|-----------------------------------------------|

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

Definition at line 937 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink_and_double()`.

4.1045.3.47 `resize()` [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
 size_type __new_size,
 const value_type & __x) [inline]
```

Resizes the vector to the specified number of elements.

## Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>__new_size</code> | Number of elements the vector should contain.     |
| <code>__x</code>        | Data with which new elements should be populated. |

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

Definition at line 957 of file `stl_vector.h`.

4.1045.3.48 `shrink_to_fit()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::shrink_to_fit () [inline]
```

A non-binding request to reduce `capacity()` to `size()`.

Definition at line 989 of file `stl_vector.h`.

#### 4.1045.3.49 `size()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
size_type std::vector< _Tp, _Alloc >::size () const [inline], [noexcept]
```

Returns the number of elements in the vector.

Definition at line 918 of file `stl_vector.h`.

Referenced by `__gnu_parallel::__shrink()`, `__gnu_parallel::__shrink_and_double()`, `std::vector< sub_match< _Bi_↵  
iter >, allocator< sub_match< _Bi_iter > > >::M_range_check()`, `__gnu_parallel::list_partition()`, `std::discrete_↵  
distribution< _IntType >::max()`, `std::operator==()`, `std::vector< sub_match< _Bi_iter >, allocator< sub_match< _↵  
Bi_iter > > >::resize()`, and `std::regex_traits< _CharType >::transform_primary()`.

#### 4.1045.3.50 `swap()`

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>>
void std::vector< _Tp, _Alloc >::swap (
 vector< _Tp, _Alloc > & __x) [inline], [noexcept]
```

Swaps data with another vector.

##### Parameters

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| <code>_↵<br/>__x</code> | A vector of the same element and allocator types. |
|-------------------------|---------------------------------------------------|

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global `std::swap()` function is specialized such that `std::swap(v1,v2)` will feed to this function.

Whether the allocators are swapped depends on the allocator traits.

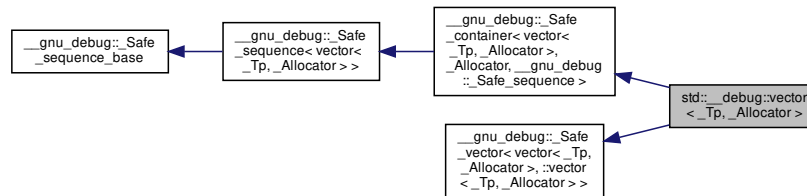
Definition at line 1480 of file `stl_vector.h`.

The documentation for this class was generated from the following files:

- [stl\\_vector.h](#)
- [vector.tcc](#)

## 4.1046 std::\_\_debug::vector&lt; \_Tp, \_Allocator &gt; Class Template Reference

Inheritance diagram for std::\_\_debug::vector< \_Tp, \_Allocator >:



## Public Types

- typedef `_Allocator` **allocator\_type**
- typedef `__gnu_debug:: Safe_iterator< _Base_const_iterator, vector >` **const\_iterator**
- typedef `_Base::const_pointer` **const\_pointer**
- typedef `_Base::const_reference` **const\_reference**
- typedef `std::reverse_iterator< const_iterator >` **const\_reverse\_iterator**
- typedef `_Base::difference_type` **difference\_type**
- typedef `__gnu_debug:: Safe_iterator< _Base_iterator, vector >` **iterator**
- typedef `_Base::pointer` **pointer**
- typedef `_Base::reference` **reference**
- typedef `std::reverse_iterator< iterator >` **reverse\_iterator**
- typedef `_Base::size_type` **size\_type**
- typedef `_Tp` **value\_type**

## Public Member Functions

- **vector** (`const _Allocator &__a`) noexcept
- **vector** (`size_type __n, const _Allocator &__a=_Allocator()`)
- **vector** (`size_type __n, const _Tp &__value, const _Allocator &__a=_Allocator()`)
- template<class `_InputIterator`, typename = `std::RequireInputIter<_InputIterator>>`  
**vector** (`_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator()`)
- **vector** (`const vector &`)=default
- **vector** (`vector &&`)=default
- **vector** (`const vector &__x, const allocator_type &__a`)
- **vector** (`vector &&__x, const allocator_type &__a`) noexcept(noexcept(`_Base`(`std::declval<_Base && >()`), `std::declval< const allocator_type & >()`))
- **vector** (`initializer_list< value_type > __l, const allocator_type &__a=allocator_type()`)
- **vector** (`const _Base &__x`)
- `_Base & _M_base` () noexcept
- `const _Base & _M_base` () const noexcept
- void `_M_invalidate_if` (`_Predicate __pred`)
- void `_M_swap` (`_Safe_container &__x`) noexcept
- void `_M_transfer_from_if` (`_Safe_sequence &__from, _Predicate __pred`)

- `template<typename _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`void assign (_InputIterator __first, _InputIterator __last)`
- `void assign (size_type __n, const _Tp &__u)`
- `void assign (initializer_list< value_type > __l)`
- `reference back () noexcept`
- `const_reference back () const noexcept`
- `iterator begin () noexcept`
- `const_iterator begin () const noexcept`
- `size_type capacity () const noexcept`
- `const_iterator cbegin () const noexcept`
- `const_iterator cend () const noexcept`
- `void clear () noexcept`
- `const_reverse_iterator crbegin () const noexcept`
- `const_reverse_iterator crend () const noexcept`
- `template<typename... _Args>`  
`iterator emplace (const_iterator __position, _Args &&... __args)`
- `template<typename... _Args>`  
`void emplace_back (_Args &&... __args)`
- `iterator end () noexcept`
- `const_iterator end () const noexcept`
- `iterator erase (const_iterator __position)`
- `iterator erase (const_iterator __first, const_iterator __last)`
- `reference front () noexcept`
- `const_reference front () const noexcept`
- `iterator insert (const_iterator __position, const _Tp &__x)`
- `template<typename _Up = _Tp>`  
`__gnu_cxx::enable_if<!std::are_same<_Up, bool>::value, iterator>::__type insert (const_iterator __position, _Tp &&__x)`
- `iterator insert (const_iterator __position, initializer_list< value_type > __l)`
- `iterator insert (const_iterator __position, size_type __n, const _Tp &__x)`
- `template<class _InputIterator, typename = std::RequireInputIter<_InputIterator>>`  
`iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)`
- `vector & operator= (const vector &)=default`
- `vector & operator= (vector &&)=default`
- `vector & operator= (initializer_list< value_type > __l)`
- `reference operator[] (size_type __n) noexcept`
- `const_reference operator[] (size_type __n) const noexcept`
- `void pop_back () noexcept`
- `void push_back (const _Tp &__x)`
- `template<typename _Up = _Tp>`  
`__gnu_cxx::enable_if<!std::are_same<_Up, bool>::value, void>::__type push_back (_Tp &&__x)`
- `reverse_iterator rbegin () noexcept`
- `const_reverse_iterator rbegin () const noexcept`
- `reverse_iterator rend () noexcept`
- `const_reverse_iterator rend () const noexcept`
- `void reserve (size_type __n)`
- `void resize (size_type __sz)`
- `void resize (size_type __sz, const _Tp &__c)`
- `void shrink_to_fit ()`
- `void swap (vector &__x) noexcept(/*conditional */)`

## Public Attributes

- `_Safe_iterator_base` \* `_M_const_iterators`
- `_Safe_iterator_base` \* `_M_iterators`
- unsigned int `_M_version`

## Protected Member Functions

- void `_M_detach_all` ()
- void `_M_detach_singular` ()
- `__gnu_cxx::__mutex` & `_M_get_mutex` () throw ()
- void `_M_invalidate_all` () const
- bool `_M_requires_reallocation` (size\_type \_\_elements) const noexcept
- void `_M_revalidate_singular` ()
- `_Safe_container` & `_M_safe` () noexcept
- void `_M_swap` (`_Safe_sequence_base` & \_\_x) noexcept
- void `_M_update_guaranteed_capacity` () noexcept

## Protected Attributes

- size\_type `_M_guaranteed_capacity`

## Friends

- `template<typename _ItT, typename _SeqT, typename _CatT>`  
class `::__gnu_debug::_Safe_iterator`

## 4.1046.1 Detailed Description

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::vector<_Tp, _Allocator>
```

Class `std::vector` with safety/checking/debug instrumentation.

Definition at line 36 of file `debug/vector`.

## 4.1046.2 Constructor &amp; Destructor Documentation

4.1046.2.1 `vector()`

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
std::__debug::vector<_Tp, _Allocator>::vector (
 const _Base & __x) [inline]
```

Construction from a normal-mode vector.

Definition at line 224 of file `debug/vector`.

### 4.1046.3 Member Function Documentation

#### 4.1046.3.1 `_M_detach_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () [protected], [inherited]
```

Detach all iterators, leaving them singular.

Referenced by `__gnu_debug::_Safe_sequence_base::~~Safe_sequence_base()`.

#### 4.1046.3.2 `_M_detach_singular()`

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () [protected], [inherited]
```

Detach all singular iterators.

#### Postcondition

for all iterators `i` attached to this sequence, `i->_M_version == _M_version`.

#### 4.1046.3.3 `_M_get_mutex()`

```
__gnu_cxx::__mutex& __gnu_debug::_Safe_sequence_base::_M_get_mutex () throw () [protected],
[inherited]
```

For use in `_Safe_sequence`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1046.3.4 `_M_invalidate_all()`

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const [inline], [protected], [inherited]
```

Invalidates all iterators.

Definition at line 256 of file `safe_base.h`.

References `__gnu_debug::_Safe_sequence_base::_M_version`.



## 4.1046.3.5 \_M\_invalidate\_if()

```
void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_invalidate_if (
 _Predicate __pred) [inherited]
```

Invalidates all iterators *x* that reference this sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 38 of file `safe_sequence.tcc`.

## 4.1046.3.6 \_M\_revalidate\_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

## 4.1046.3.7 \_M\_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
 _Safe_sequence_base & __x) [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

## 4.1046.3.8 \_M\_transfer\_from\_if()

```
void __gnu_debug::_Safe_sequence< vector< _Tp, _Allocator > >::_M_transfer_from_if (
 _Safe_sequence< vector< _Tp, _Allocator > > & __from,
 _Predicate __pred) [inherited]
```

Transfers all iterators *x* that reference *from* sequence, are not singular, and for which `__pred(x)` returns true. `__pred` will be invoked with the normal iterators nested in the safe ones.

Definition at line 69 of file `safe_sequence.tcc`.

## 4.1046.4 Member Data Documentation

## 4.1046.4.1 \_M\_const\_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [inherited]
```

The list of constant iterators that reference this container.

Definition at line 197 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1046.4.2 `_M_iterators`

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [inherited]
```

The list of mutable iterators that reference this container.

Definition at line 194 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence< map< _Key, _Tp, _Compare, _Allocator > >::_M_transfer_from_if()`.

#### 4.1046.4.3 `_M_version`

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
```

The container version number. This number may never be 0.

Definition at line 200 of file `safe_base.h`.

Referenced by `__gnu_debug::_Safe_sequence_base::_M_invalidate_all()`.

The documentation for this class was generated from the following file:

- [debug/vector](#)

### 4.1047 `std::vector< bool, _Alloc >` Class Template Reference

Inherits `std::_Bvector_base< _Alloc >`.

#### Public Types

- typedef `_Alloc` **allocator\_type**
- typedef `_Bit_const_iterator` **const\_iterator**
- typedef `const bool *` **const\_pointer**
- typedef `bool` **const\_reference**
- typedef [std::reverse\\_iterator](#)< `const_iterator` > **const\_reverse\_iterator**
- typedef `ptrdiff_t` **difference\_type**
- typedef `_Bit_iterator` **iterator**
- typedef `_Bit_reference *` **pointer**
- typedef `_Bit_reference` **reference**
- typedef [std::reverse\\_iterator](#)< `iterator` > **reverse\_iterator**
- typedef `size_t` **size\_type**
- typedef `bool` **value\_type**

## Public Member Functions

- **vector** (const allocator\_type &\_\_a)
- **vector** (size\_type \_\_n, const allocator\_type &\_\_a=allocator\_type())
- **vector** (size\_type \_\_n, const bool &\_\_value, const allocator\_type &\_\_a=allocator\_type())
- **vector** (const [vector](#) &\_\_x)
- **vector** ([vector](#) &&)=default
- **vector** ([vector](#) &&\_\_x, const allocator\_type &\_\_a) noexcept(\_Bit\_alloc\_traits::\_S\_always\_equal())
- **vector** (const [vector](#) &\_\_x, const allocator\_type &\_\_a)
- **vector** ([initializer\\_list](#)< bool > \_\_l, const allocator\_type &\_\_a=allocator\_type())
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
**vector** (\_InputIterator \_\_first, \_InputIterator \_\_last, const allocator\_type &\_\_a=allocator\_type())
- void **assign** (size\_type \_\_n, const bool &\_\_x)
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
void **assign** (\_InputIterator \_\_first, \_InputIterator \_\_last)
- void **assign** ([initializer\\_list](#)< bool > \_\_l)
- reference **at** (size\_type \_\_n)
- const\_reference **at** (size\_type \_\_n) const
- reference **back** ()
- const\_reference **back** () const
- iterator **begin** () noexcept
- const\_iterator **begin** () const noexcept
- size\_type **capacity** () const noexcept
- const\_iterator **cbegin** () const noexcept
- const\_iterator **cend** () const noexcept
- void **clear** () noexcept
- [const\\_reverse\\_iterator](#) **crbegin** () const noexcept
- [const\\_reverse\\_iterator](#) **crend** () const noexcept
- void **data** () noexcept
- template<typename... \_Args>  
iterator **emplace** (const\_iterator \_\_pos, \_Args &&... \_\_args)
- template<typename... \_Args>  
void **emplace\_back** (\_Args &&... \_\_args)
- bool **empty** () const noexcept
- iterator **end** () noexcept
- const\_iterator **end** () const noexcept
- iterator **erase** (const\_iterator \_\_position)
- iterator **erase** (const\_iterator \_\_first, const\_iterator \_\_last)
- void **flip** () noexcept
- reference **front** ()
- const\_reference **front** () const
- allocator\_type **get\_allocator** () const
- iterator **insert** (const\_iterator \_\_position, const bool &\_\_x=bool())
- template<typename \_InputIterator, typename = std::::RequireInputIter<\_InputIterator>>  
iterator **insert** (const\_iterator \_\_position, \_InputIterator \_\_first, \_InputIterator \_\_last)
- iterator **insert** (const\_iterator \_\_position, size\_type \_\_n, const bool &\_\_x)
- iterator **insert** (const\_iterator \_\_p, [initializer\\_list](#)< bool > \_\_l)
- size\_type **max\_size** () const noexcept
- [vector](#) & **operator=** (const [vector](#) &\_\_x)
- [vector](#) & **operator=** ([vector](#) &&\_\_x) noexcept(\_Bit\_alloc\_traits::\_S\_nothrow\_move())
- [vector](#) & **operator=** ([initializer\\_list](#)< bool > \_\_l)

- reference **operator[]** (size\_type \_\_n)
- const\_reference **operator[]** (size\_type \_\_n) const
- void **pop\_back** ()
- void **push\_back** (bool \_\_x)
- [reverse\\_iterator](#) **rbegin** () noexcept
- [const\\_reverse\\_iterator](#) **rbegin** () const noexcept
- [reverse\\_iterator](#) **rend** () noexcept
- [const\\_reverse\\_iterator](#) **rend** () const noexcept
- void **reserve** (size\_type \_\_n)
- void **resize** (size\_type \_\_new\_size, bool \_\_x=bool())
- void **shrink\_to\_fit** ()
- size\_type **size** () const noexcept
- void **swap** ([vector](#) &\_\_x) noexcept

#### Static Public Member Functions

- static void **swap** (reference \_\_x, reference \_\_y) noexcept

#### Protected Types

- typedef [\\_\\_gnu\\_cxx::\\_\\_alloc\\_traits](#)<\_Alloc >::template rebind<\_Bit\_type >::other **\_Bit\_alloc\_type**

#### Protected Member Functions

- \_Bit\_pointer **\_M\_allocate** (size\_t \_\_n)
- template<typename \_InputIterator >  
void **\_M\_assign\_aux** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_assign\_aux** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))
- size\_type **\_M\_check\_len** (size\_type \_\_n, const char \* \_\_s) const
- iterator **\_M\_copy\_aligned** (const\_iterator \_\_first, const\_iterator \_\_last, iterator \_\_result)
- void **\_M\_deallocate** ()
- iterator **\_M\_erase** (iterator \_\_pos)
- iterator **\_M\_erase** (iterator \_\_first, iterator \_\_last)
- void **\_M\_erase\_at\_end** (iterator \_\_pos)
- void **\_M\_fill\_assign** (size\_t \_\_n, bool \_\_x)
- void **\_M\_fill\_insert** (iterator \_\_position, size\_type \_\_n, bool \_\_x)
- \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () noexcept
- const \_Bit\_alloc\_type & **\_M\_get\_Bit\_allocator** () const noexcept
- void **\_M\_initialize** (size\_type \_\_n)
- template<typename \_Integer >  
void **\_M\_initialize\_dispatch** (\_Integer \_\_n, \_Integer \_\_x, \_\_true\_type)
- template<typename \_InputIterator >  
void **\_M\_initialize\_dispatch** (\_InputIterator \_\_first, \_InputIterator \_\_last, \_\_false\_type)
- template<typename \_InputIterator >  
void **\_M\_initialize\_range** (\_InputIterator \_\_first, \_InputIterator \_\_last, [std::input\\_iterator\\_tag](#))
- template<typename \_ForwardIterator >  
void **\_M\_initialize\_range** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::forward\\_iterator\\_tag](#))

- `void _M_initialize_value (bool __x)`
- `void _M_insert_aux (iterator __position, bool __x)`
- `template<typename _Integer >`  
`void _M_insert_dispatch (iterator __pos, _Integer __n, _Integer __x, __true_type)`
- `template<typename _InputIterator >`  
`void _M_insert_dispatch (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)`
- `template<typename _InputIterator >`  
`void _M_insert_range (iterator __pos, _InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`void _M_insert_range (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `void _M_move_data (_Bvector_base &&__x) noexcept`
- `void _M_range_check (size_type __n) const`
- `void _M_reallocate (size_type __n)`
- `bool _M_shrink_to_fit ()`

#### Static Protected Member Functions

- `static size_t _S_nword (size_t __n)`

#### Protected Attributes

- `_Bvector_impl _M_impl`

#### Friends

- `struct std::hash< vector >`

#### 4.1047.1 Detailed Description

```
template<typename _Alloc>
class std::vector< bool, _Alloc >
```

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

#### Template Parameters

|                     |                 |
|---------------------|-----------------|
| <code>_Alloc</code> | Allocator type. |
|---------------------|-----------------|

Note that `vector<bool>` does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to `bool`. See DR96 for details.

#### See also

[vector](#) for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ( `[]` ) access is also provided as with C-style arrays.

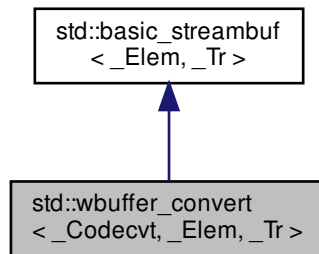
Definition at line 615 of file `stl_bvector.h`.

The documentation for this class was generated from the following files:

- [stl\\_bvector.h](#)
- [vector.tcc](#)

#### 4.1048 `std::wbuffer_convert<_Codecvt, _Elem, _Tr>` Class Template Reference

Inheritance diagram for `std::wbuffer_convert<_Codecvt, _Elem, _Tr>`:



##### Public Types

- `typedef _Codecvt::state_type` **state\_type**
- `typedef _Elem` [char\\_type](#)
- `typedef _Tr` **traits\_type**
- `typedef traits_type::int_type` **int\_type**
- `typedef traits_type::pos_type` **pos\_type**
- `typedef traits_type::off_type` **off\_type**
- `typedef` [basic\\_streambuf](#)[< char\\_type, traits\\_type >](#) [\\_\\_streambuf\\_type](#)

## Public Member Functions

- [wbuffer\\_convert](#) ()
  - [wbuffer\\_convert](#) ([streambuf](#) \* \_\_bytebuf, [\\_Codecvt](#) \* \_\_pcvt=new [\\_Codecvt](#), [state\\_type](#) \_\_state=[state\\_type](#)())
  - [wbuffer\\_convert](#) (const [wbuffer\\_convert](#) &)=delete
  - [locale](#) [getloc](#) () const
  - [streamsize](#) [in\\_avail](#) ()
  - [wbuffer\\_convert](#) & [operator=](#) (const [wbuffer\\_convert](#) &)=delete
  - [locale](#) [pubimbue](#) (const [locale](#) & \_\_loc)
  - [streambuf](#) \* [rdbuf](#) () const noexcept
  - [streambuf](#) \* [rdbuf](#) ([streambuf](#) \* \_\_bytebuf) noexcept
  - [int\\_type](#) [sbumpc](#) ()
  - [int\\_type](#) [sgetc](#) ()
  - [streamsize](#) [sgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - [int\\_type](#) [snextc](#) ()
  - [int\\_type](#) [sputbackc](#) ([char\\_type](#) \_\_c)
  - [int\\_type](#) [sputc](#) ([char\\_type](#) \_\_c)
  - [streamsize](#) [sputn](#) (const [char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - [state\\_type](#) [state](#) () const noexcept
  - [int\\_type](#) [sungetc](#) ()
- 
- [basic\\_streambuf](#) \* [pubsetbuf](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
  - [pos\\_type](#) [pubseekoff](#) ([off\\_type](#) \_\_off, [ios\\_base::seekdir](#) \_\_way, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - [pos\\_type](#) [pubseekpos](#) ([pos\\_type](#) \_\_sp, [ios\\_base::openmode](#) \_\_mode=[ios\\_base::in](#)|[ios\\_base::out](#))
  - [int](#) [pubsync](#) ()

## Protected Member Functions

- void [\\_\\_safe\\_gbump](#) ([streamsize](#) \_\_n)
- void [\\_\\_safe\\_pbump](#) ([streamsize](#) \_\_n)
- void [gbump](#) (int \_\_n)
- virtual void [imbue](#) (const [locale](#) & \_\_loc)
- [\\_Wide\\_streambuf::int\\_type](#) [overflow](#) (typename [\\_Wide\\_streambuf::int\\_type](#) \_\_out)
- virtual [int\\_type](#) [pbackfail](#) ([int\\_type](#) \_\_c=traits\_type::eof())
- void [pbump](#) (int \_\_n)
- virtual [pos\\_type](#) [seekoff](#) ([off\\_type](#), [ios\\_base::seekdir](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [pos\\_type](#) [seekpos](#) ([pos\\_type](#), [ios\\_base::openmode](#)=[ios\\_base::in](#)|[ios\\_base::out](#))
- virtual [basic\\_streambuf](#)< [char\\_type](#), [\\_Tr](#) > \* [setbuf](#) ([char\\_type](#) \*, [streamsize](#))
- void [setg](#) ([char\\_type](#) \* \_\_gbeg, [char\\_type](#) \* \_\_gnext, [char\\_type](#) \* \_\_gend)
- void [setp](#) ([char\\_type](#) \* \_\_pbeg, [char\\_type](#) \* \_\_pend)
- virtual [streamsize](#) [showmanyc](#) ()
- void [swap](#) ([basic\\_streambuf](#) & \_\_sb)
- [int](#) [sync](#) ()
- virtual [int\\_type](#) [uflow](#) ()
- [\\_Wide\\_streambuf::int\\_type](#) [underflow](#) ()
- virtual [streamsize](#) [xsgetn](#) ([char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)
- [streamsize](#) [xspurn](#) (const typename [\\_Wide\\_streambuf::char\\_type](#) \* \_\_s, [streamsize](#) \_\_n)

- virtual `streamsize xsputn` (const `char_type` \*\_\_s, `streamsize` \_\_n)

- `char_type` \* `eback` () const
- `char_type` \* `gptr` () const
- `char_type` \* `egptr` () const

- `char_type` \* `pbase` () const
- `char_type` \* `pptr` () const
- `char_type` \* `epptr` () const

#### Protected Attributes

- `locale` `_M_buf_locale`
- `char_type` \* `_M_in_beg`
- `char_type` \* `_M_in_cur`
- `char_type` \* `_M_in_end`
- `char_type` \* `_M_out_beg`
- `char_type` \* `_M_out_cur`
- `char_type` \* `_M_out_end`

#### 4.1048.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
```

Buffer conversions.

Definition at line 387 of file `locale_conv.h`.

#### 4.1048.2 Member Typedef Documentation

##### 4.1048.2.1 `__streambuf_type`

```
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _Elem , _Tr >::__streambuf_type
[inherited]
```

This is a non-standard type.

Definition at line 140 of file `streambuf`.



#### 4.1048.2.2 `char_type`

```
typedef _Elem std::basic_streambuf< _Elem , _Tr >::char_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 131 of file streambuf.

#### 4.1048.2.3 `int_type`

```
typedef traits_type::int_type std::basic_streambuf< _Elem , _Tr >::int_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 133 of file streambuf.

#### 4.1048.2.4 `off_type`

```
typedef traits_type::off_type std::basic_streambuf< _Elem , _Tr >::off_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 135 of file streambuf.

#### 4.1048.2.5 `pos_type`

```
typedef traits_type::pos_type std::basic_streambuf< _Elem , _Tr >::pos_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 134 of file streambuf.

#### 4.1048.2.6 `traits_type`

```
typedef _Tr std::basic_streambuf< _Elem , _Tr >::traits_type [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

Definition at line 132 of file streambuf.

#### 4.1048.3 Constructor & Destructor Documentation

##### 4.1048.3.1 `wbuffer_convert()` [1/2]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert () [inline]
```

Default constructor.

Definition at line 395 of file `locale_conv.h`.

##### 4.1048.3.2 `wbuffer_convert()` [2/2]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
 streambuf * __bytebuf,
 _Codecvt * __pcvt = new _Codecvt,
 state_type __state = state_type()) [inline], [explicit]
```

Constructor.

###### Parameters

|                        |                                    |
|------------------------|------------------------------------|
| <code>__bytebuf</code> | The underlying byte stream buffer. |
| <code>__pcvt</code>    | The facet to use for conversions.  |
| <code>__state</code>   | Initial conversion state.          |

Takes ownership of `__pcvt` and will delete it in the destructor.

Definition at line 406 of file `locale_conv.h`.

References `std::basic_streambuf< _Elem, _Tr >::setg()`, and `std::basic_streambuf< _Elem, _Tr >::setp()`.

#### 4.1048.4 Member Function Documentation

##### 4.1048.4.1 `eback()`

```
char_type* std::basic_streambuf< _Elem , _Tr >::eback () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 489 of file `streambuf`.

#### 4.1048.4.2 `egptr()`

```
char_type* std::basic_streambuf<_Elem, _Tr>::egptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 495 of file `streambuf`.

#### 4.1048.4.3 `epptr()`

```
char_type* std::basic_streambuf<_Elem, _Tr>::epptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- `pbase()` returns the beginning pointer for the output sequence
- `pptr()` returns the next pointer for the output sequence
- `epptr()` returns the end pointer for the output sequence

Definition at line 542 of file `streambuf`.

#### 4.1048.4.4 `gbump()`

```
void std::basic_streambuf<_Elem, _Tr>::gbump (
 int __n) [inline], [protected], [inherited]
```

Moving the read position.

**Parameters**

|                 |                             |
|-----------------|-----------------------------|
| <code>_↵</code> | The delta by which to move. |
| <code>_n</code> |                             |

This just advances the read position without returning any data.

Definition at line 505 of file streambuf.

**4.1048.4.5 getloc()**

```
locale std::basic_streambuf< _Elem , _Tr >::getloc () const [inline], [inherited]
```

Locale access.

**Returns**

The current locale in effect.

If `pubimbue(loc)` has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

Definition at line 233 of file streambuf.

**4.1048.4.6 gptr()**

```
char_type* std::basic_streambuf< _Elem , _Tr >::gptr () const [inline], [protected], [inherited]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- `eback()` returns the beginning pointer for the input sequence
- `gptr()` returns the next pointer for the input sequence
- `egptr()` returns the end pointer for the input sequence

Definition at line 492 of file streambuf.

**4.1048.4.7 imbue()**

```
virtual void std::basic_streambuf< _Elem , _Tr >::imbue (
 const locale & __loc) [inline], [protected], [virtual], [inherited]
```

Changes translations.

## Parameters

|                    |               |
|--------------------|---------------|
| <code>__loc</code> | A new locale. |
|--------------------|---------------|

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

## Note

Base class version does nothing.

Definition at line 583 of file streambuf.

## 4.1048.4.8 in\_avail()

```
streamsize std::basic_streambuf<_Elem, _Tr>::in_avail () [inline], [inherited]
```

Looking ahead into the stream.

## Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

Definition at line 291 of file streambuf.

## 4.1048.4.9 overflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert<_Codecvt, _Elem, _Tr>::overflow (
 typename _Wide_streambuf::int_type __c) [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

## Parameters

|                  |                                     |
|------------------|-------------------------------------|
| <code>__c</code> | An additional character to consume. |
|------------------|-------------------------------------|

**Returns**

`eof()` to indicate failure, something else (usually `__c`, or `not_eof()`)

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character `c` is also written out, if `__c` is not `eof()`.

For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.

A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

**Note**

Base class version does nothing, returns `eof()`.

Reimplemented from [std::basic\\_streambuf<\\_Elem, \\_Tr>](#).

Definition at line 450 of file `locale_conv.h`.

References [std::basic\\_streambuf<\\_Elem, \\_Tr>::sputc\(\)](#).

**4.1048.4.10 pbackfail()**

```
virtual int_type std::basic_streambuf<_Elem, _Tr>::pbackfail (
 int_type __c = traits_type::eof()) [inline], [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

**Parameters**

|                  |                                                      |
|------------------|------------------------------------------------------|
| <code>__c</code> | The character to be inserted back into the sequence. |
|------------------|------------------------------------------------------|

**Returns**

`eof()` on failure, *some other value* on success

**Postcondition**

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

**Note**

Base class version does nothing, returns `eof()`.

Definition at line 731 of file `streambuf`.

## 4.1048.4.11 pbase()

```
char_type* std::basic_streambuf<_Elem, _Tr>::pbase () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 536 of file streambuf.

## 4.1048.4.12 pbump()

```
void std::basic_streambuf<_Elem, _Tr>::pbump (
 int __n) [inline], [protected], [inherited]
```

Moving the write position.

## Parameters

|                  |                             |
|------------------|-----------------------------|
| <code>__n</code> | The delta by which to move. |
|------------------|-----------------------------|

This just advances the write position without returning any data.

Definition at line 552 of file streambuf.

## 4.1048.4.13 pptr()

```
char_type* std::basic_streambuf<_Elem, _Tr>::pptr () const [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Definition at line 539 of file streambuf.

#### 4.1048.4.14 pubimbue()

```
locale std::basic_streambuf< _Elem , _Tr >::pubimbue (
 const locale & __loc) [inline], [inherited]
```

Entry point for imbue().

##### Parameters

|                    |                 |
|--------------------|-----------------|
| <code>__loc</code> | The new locale. |
|--------------------|-----------------|

##### Returns

The previous locale.

Calls the derived imbue(\_\_loc).

Definition at line 216 of file streambuf.

#### 4.1048.4.15 pubseekoff()

```
pos_type std::basic_streambuf< _Elem , _Tr >::pubseekoff (
 off_type __off,
 ios_base::seekdir __way,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.

##### Parameters

|                     |                               |
|---------------------|-------------------------------|
| <code>__off</code>  | Offset.                       |
| <code>__way</code>  | Value for ios_base::seekdir.  |
| <code>__mode</code> | Value for ios_base::openmode. |

Calls virtual seekoff function.

Definition at line 258 of file streambuf.

#### 4.1048.4.16 pubseekpos()

```
pos_type std::basic_streambuf< _Elem , _Tr >::pubseekpos (
 pos_type __sp,
 ios_base::openmode __mode = ios_base::in | ios_base::out) [inline], [inherited]
```

Alters the stream position.



## Parameters

|                     |                                             |
|---------------------|---------------------------------------------|
| <code>__sp</code>   | Position                                    |
| <code>__mode</code> | Value for <code>ios_base::openmode</code> . |

Calls virtual `seekpos` function.

Definition at line 270 of file `streambuf`.

4.1048.4.17 `pubsetbuf()`

```
basic_streambuf* std::basic_streambuf<_Elem, _Tr>::pubsetbuf (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry points for derived buffer functions.

The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

Definition at line 246 of file `streambuf`.

4.1048.4.18 `pubsync()`

```
int std::basic_streambuf<_Elem, _Tr>::pubsync () [inline], [inherited]
```

Calls virtual `sync` function.

Definition at line 278 of file `streambuf`.

4.1048.4.19 `sbumpc()`

```
int_type std::basic_streambuf<_Elem, _Tr>::sbumpc () [inline], [inherited]
```

Getting the next character.

## Returns

The next character, or `eof`.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

Definition at line 323 of file `streambuf`.

#### 4.1048.4.20 seekoff()

```
virtual pos_type std::basic_streambuf< _Elem , _Tr >::seekoff (
 off_type ,
 ios_base::seekdir ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 609 of file `streambuf`.

#### 4.1048.4.21 seekpos()

```
virtual pos_type std::basic_streambuf< _Elem , _Tr >::seekpos (
 pos_type ,
 ios_base::openmode = ios_base::in | ios_base::out) [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.

Each derived class provides its own appropriate behavior.

##### Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Definition at line 621 of file `streambuf`.

#### 4.1048.4.22 setbuf()

```
virtual basic_streambuf<char_type,_Tr >* std::basic_streambuf< _Elem , _Tr >::setbuf (
 char_type * ,
 streamsize) [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.

Each derived class provides its own appropriate behavior. See the next-to-last paragraph of <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering> for more on this function.

##### Note

Base class version does nothing, returns `this`.

Definition at line 598 of file `streambuf`.

## 4.1048.4.23 setg()

```
void std::basic_streambuf<_Elem, _Tr>::setg (
 char_type * __gbeg,
 char_type * __gnext,
 char_type * __gend) [inline], [protected], [inherited]
```

Setting the three read area pointers.

## Parameters

|                      |            |
|----------------------|------------|
| <code>__gbeg</code>  | A pointer. |
| <code>__gnext</code> | A pointer. |
| <code>__gend</code>  | A pointer. |

## Postcondition

`__gbeg == eback()`, `__gnext == gptr()`, and `__gend == egptr()`

Definition at line 516 of file streambuf.

## 4.1048.4.24 setp()

```
void std::basic_streambuf<_Elem, _Tr>::setp (
 char_type * __pbeg,
 char_type * __pend) [inline], [protected], [inherited]
```

Setting the three write area pointers.

## Parameters

|                     |            |
|---------------------|------------|
| <code>__pbeg</code> | A pointer. |
| <code>__pend</code> | A pointer. |

## Postcondition

`__pbeg == pbase()`, `__pbeg == pptr()`, and `__pend == epptr()`

Definition at line 562 of file streambuf.

## 4.1048.4.25 sgetc()

```
int_type std::basic_streambuf<_Elem, _Tr>::sgetc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

Definition at line 345 of file `streambuf`.

**4.1048.4.26 `sgetn()`**

```
streamsize std::basic_streambuf< _Elem , _Tr >::sgetn (
 char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for `xsgetn`.

**Parameters**

|                  |                |
|------------------|----------------|
| <code>__s</code> | A buffer area. |
| <code>__n</code> | A count.       |

Returns `xsgetn(__s,__n)`. The effect is to fill `__s[0]` through `__s[__n-1]` with characters from the input sequence, if possible.

Definition at line 364 of file `streambuf`.

**4.1048.4.27 `showmanyc()`**

```
virtual streamsize std::basic_streambuf< _Elem , _Tr >::showmanyc () [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

**Returns**

An estimate of the number of characters available in the input sequence, or -1.

*If it returns a positive value, then successive calls to `underflow()` will not return `traits::eof()` until at least that number of characters have been supplied. If `showmanyc()` returns -1, then calls to `underflow()` or `uflow()` will fail. [27.5.2.4.3]/1*

**Note**

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to `underflow` or `uflow`] will not return `eof()` but that they will return immediately.*

The standard adds that *the morphemes of `showmanyc` are **es-how-many-see**, not **show-manic**.*

Definition at line 656 of file `streambuf`.

## 4.1048.4.28 snextc()

```
int_type std::basic_streambuf<_Elem, _Tr>::snextc () [inline], [inherited]
```

Getting the next character.

**Returns**

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

Definition at line 305 of file `streambuf`.

## 4.1048.4.29 sputbackc()

```
int_type std::basic_streambuf<_Elem, _Tr>::sputbackc (
 char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

**Parameters**

|                  |                             |
|------------------|-----------------------------|
| <code>__c</code> | The character to push back. |
|------------------|-----------------------------|

**Returns**

The previous character, if possible.

Similar to `sungetc()`, but `__c` is pushed onto the stream instead of *the previous character*. If successful, the next character fetched from the input stream will be `__c`.

Definition at line 379 of file `streambuf`.

## 4.1048.4.30 sputc()

```
int_type std::basic_streambuf<_Elem, _Tr>::sputc (
 char_type __c) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                        |
|------------------|------------------------|
| <code>__c</code> | A character to output. |
|------------------|------------------------|

**Returns**

`__c`, if possible.

One of two public output functions.

If a write position is available for the output sequence (i.e., the buffer is not full), stores `__c` in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(__c)`.

Definition at line 431 of file `streambuf`.

**4.1048.4.31 sputn()**

```
streamsize std::basic_streambuf<_Elem, _Tr>::sputn (
 const char_type * __s,
 streamsize __n) [inline], [inherited]
```

Entry point for all single-character output functions.

**Parameters**

|                  |                     |
|------------------|---------------------|
| <code>__s</code> | A buffer read area. |
| <code>__n</code> | A count.            |

One of two public output functions.

Returns `xsgputn(__s, __n)`. The effect is to write `__s[0]` through `__s[__n-1]` to the output sequence, if possible.

Definition at line 457 of file `streambuf`.

**4.1048.4.32 state()**

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert<_Codecvt, _Elem, _Tr>::state () const [inline], [noexcept]
```

The conversion state following the last conversion.

Definition at line 442 of file `locale_conv.h`.

## 4.1048.4.33 sungetc()

```
int_type std::basic_streambuf< _Elem, _Tr >::sungetc () [inline], [inherited]
```

Moving backwards in the input stream.

**Returns**

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

Definition at line 404 of file streambuf.

## 4.1048.4.34 sync()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync (
 void) [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

**Returns**

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

**Note**

Base class version does nothing, returns zero.

Reimplemented from `std::basic_streambuf< _Elem, _Tr >`.

Definition at line 446 of file locale\_conv.h.

References `std::basic_streambuf< _CharT, _Traits >::pubsync()`.

#### 4.1048.4.35 uflow()

```
virtual int_type std::basic_streambuf< _Elem , _Tr >::uflow () [inline], [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

Definition at line 707 of file `streambuf`.

#### 4.1048.4.36 underflow()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow () [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

##### Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.

For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.

A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see <https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html>

##### Note

Base class version does nothing, returns `eof()`.

Reimplemented from `std::basic_streambuf< _Elem, _Tr >`.

Definition at line 460 of file `locale_conv.h`.

References `std::basic_streambuf< _Elem, _Tr >::egptr()`, and `std::basic_streambuf< _Elem, _Tr >::gptr()`.

#### 4.1048.4.37 xsgetn()

```
streamsize std::basic_streambuf< _Elem , _Tr >::xsgetn (
 char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character extraction.



## Parameters

|                  |                                         |
|------------------|-----------------------------------------|
| <code>__s</code> | A buffer area.                          |
| <code>__n</code> | Maximum number of characters to assign. |

## Returns

The number of characters assigned.

Fills `__s[0]` through `__s[__n-1]` with characters from the input sequence, as if by `sbumpc()`. Stops when either `__n` characters have been copied, or when `traits::eof()` would be copied.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Definition at line 46 of file `streambuf.tcc`.

4.1048.4.38 `xspn()`

```
streamsize std::basic_streambuf<_Elem, _Tr>::xspn (
 const char_type * __s,
 streamsize __n) [protected], [virtual], [inherited]
```

Multiple character insertion.

## Parameters

|                  |                                        |
|------------------|----------------------------------------|
| <code>__s</code> | A buffer area.                         |
| <code>__n</code> | Maximum number of characters to write. |

## Returns

The number of characters written.

Writes `__s[0]` through `__s[__n-1]` to the output sequence, as if by `sputc()`. Stops when either `n` characters have been copied, or when `sputc()` would return `traits::eof()`.

It is expected that derived classes provide a more efficient implementation by overriding this definition.

Definition at line 80 of file `streambuf.tcc`.

## 4.1048.5 Member Data Documentation

#### 4.1048.5.1 `_M_buf_locale`

`locale std::basic_streambuf< _Elem , _Tr >::_M_buf_locale` [protected], [inherited]

Current locale setting.

Definition at line 199 of file `streambuf`.

#### 4.1048.5.2 `_M_in_beg`

`char_type* std::basic_streambuf< _Elem , _Tr >::_M_in_beg` [protected], [inherited]

Start of get area.

Definition at line 191 of file `streambuf`.

#### 4.1048.5.3 `_M_in_cur`

`char_type* std::basic_streambuf< _Elem , _Tr >::_M_in_cur` [protected], [inherited]

Current read area.

Definition at line 192 of file `streambuf`.

#### 4.1048.5.4 `_M_in_end`

`char_type* std::basic_streambuf< _Elem , _Tr >::_M_in_end` [protected], [inherited]

End of get area.

Definition at line 193 of file `streambuf`.

#### 4.1048.5.5 `_M_out_beg`

`char_type* std::basic_streambuf< _Elem , _Tr >::_M_out_beg` [protected], [inherited]

Start of put area.

Definition at line 194 of file `streambuf`.

4.1048.5.6 `_M_out_cur`

```
char_type* std::basic_streambuf<_Elem, _Tr>::_M_out_cur [protected], [inherited]
```

Current put area.

Definition at line 195 of file `streambuf`.

4.1048.5.7 `_M_out_end`

```
char_type* std::basic_streambuf<_Elem, _Tr>::_M_out_end [protected], [inherited]
```

End of put area.

Definition at line 196 of file `streambuf`.

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

4.1049 `std::weak_ptr<_Tp>` Class Template Reference

Inherits `std::__weak_ptr<_Tp, _Lp>`.

## Public Types

- using **element\_type** = typename [remove\\_extent](#)<\_Tp>::type

## Public Member Functions

- `template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>`  
**weak\_ptr** (const [shared\\_ptr](#)<\_Yp> &\_\_r) noexcept
- **weak\_ptr** (const [weak\\_ptr](#) &) noexcept=default
- `template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>`  
**weak\_ptr** (const [weak\\_ptr](#)<\_Yp> &\_\_r) noexcept
- **weak\_ptr** ([weak\\_ptr](#) &&) noexcept=default
- `template<typename _Yp, typename = _Constructible<weak_ptr<_Yp>>>`  
**weak\_ptr** ([weak\\_ptr](#)<\_Yp> &&\_\_r) noexcept
- `bool expired ()` const noexcept
- `shared\_ptr<_Tp> lock ()` const noexcept
- `weak\_ptr & operator= (const weak\_ptr &__r)` noexcept=default
- `template<typename _Yp>`  
`_Assignable< const weak\_ptr<_Yp> &> operator= (const weak\_ptr<_Yp> &__r)` noexcept
- `template<typename _Yp>`  
`_Assignable< const shared\_ptr<_Yp> &> operator= (const shared\_ptr<_Yp> &__r)` noexcept
- `weak\_ptr & operator= (weak\_ptr &&__r)` noexcept=default
- `template<typename _Yp>`  
`_Assignable< weak\_ptr<_Yp> > operator= (weak\_ptr<_Yp> &&__r)` noexcept
- `template<typename _Tp1>`  
`bool owner_before (const __shared_ptr<_Tp1, _Lp> &__rhs)` const noexcept
- `template<typename _Tp1>`  
`bool owner_before (const __weak_ptr<_Tp1, _Lp> &__rhs)` const noexcept
- `void reset ()` noexcept
- `void swap (__weak_ptr &__s)` noexcept
- `long use_count ()` const noexcept

## Related Functions

(Note that these are not member functions.)

- `template<typename _Tp >`  
`void swap (weak\_ptr<_Tp> &__a, weak\_ptr<_Tp> &__b) noexcept`

### 4.1049.1 Detailed Description

```
template<typename _Tp>
class std::weak_ptr<_Tp>
```

A non-owning observer for a pointer owned by a `shared_ptr`.

A `weak_ptr` provides a safe alternative to a raw pointer when you want a non-owning reference to an object that is managed by a `shared_ptr`.

Unlike a raw pointer, a `weak_ptr` can be converted to a new `shared_ptr` that shares ownership with every other `shared_ptr` that already owns the pointer. In other words you can upgrade from a non-owning "weak" reference to an owning `shared_ptr`, without having access to any of the existing `shared_ptr` objects.

Also unlike a raw pointer, a `weak_ptr` does not become "dangling" after the object it points to has been destroyed. Instead, a `weak_ptr` becomes *expired* and can no longer be converted to a `shared_ptr` that owns the freed pointer, so you cannot accidentally access the pointed-to object after it has been destroyed.

Definition at line 685 of file `bits/shared_ptr.h`.

The documentation for this class was generated from the following file:

- [bits/shared\\_ptr.h](#)

## 4.1050 `std::weibull_distribution<_RealType>` Class Template Reference

### Classes

- struct [param\\_type](#)

### Public Types

- typedef `_RealType` [result\\_type](#)

## Public Member Functions

- **weibull\_distribution** (\_RealType \_\_a, \_RealType \_\_b=\_RealType(1))
- **weibull\_distribution** (const [param\\_type](#) &\_\_p)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_ForwardIterator, typename \_UniformRandomNumberGenerator >  
void **\_\_generate** (\_ForwardIterator \_\_f, \_ForwardIterator \_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- template<typename \_UniformRandomNumberGenerator >  
void **\_\_generate** ([result\\_type](#) \*\_\_f, [result\\_type](#) \*\_\_t, \_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- \_RealType **a** () const
- \_RealType **b** () const
- [result\\_type](#) **max** () const
- [result\\_type](#) **min** () const
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng)
- template<typename \_UniformRandomNumberGenerator >  
[result\\_type](#) **operator()** (\_UniformRandomNumberGenerator &\_\_urng, const [param\\_type](#) &\_\_p)
- [param\\_type](#) **param** () const
- void **param** (const [param\\_type](#) &\_\_param)
- void **reset** ()

## Friends

- bool **operator==** (const [weibull\\_distribution](#) &\_\_d1, const [weibull\\_distribution](#) &\_\_d2)

## 4.1050.1 Detailed Description

```
template<typename _RealType = double>
class std::weibull_distribution<_RealType>
```

A weibull\_distribution random number distribution.

The formula for the normal probability density function is:

$$p(x|\alpha, \beta) = \frac{\alpha}{\beta} \left(\frac{x}{\beta}\right)^{\alpha-1} \exp\left(-\left(\frac{x}{\beta}\right)^\alpha\right)$$

Definition at line 4860 of file random.h.

## 4.1050.2 Member Typedef Documentation

#### 4.1050.2.1 `result_type`

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

Definition at line 4863 of file random.h.

#### 4.1050.3 Member Function Documentation

##### 4.1050.3.1 `a()`

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a () const [inline]
```

Return the  $a$  parameter of the distribution.

Definition at line 4925 of file random.h.

##### 4.1050.3.2 `b()`

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::b () const [inline]
```

Return the  $b$  parameter of the distribution.

Definition at line 4932 of file random.h.

##### 4.1050.3.3 `max()`

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::max () const [inline]
```

Returns the least upper bound value of the distribution.

Definition at line 4961 of file random.h.

References `std::numeric_limits<_Tp>::max()`.

4.1050.3.4 `min()`

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::min () const [inline]
```

Returns the greatest lower bound value of the distribution.

Definition at line 4954 of file random.h.

4.1050.3.5 `operator()()`

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator >
result_type std::weibull_distribution< _RealType >::operator() (
 _UniformRandomNumberGenerator & __urng) [inline]
```

Generating functions.

Definition at line 4969 of file random.h.

4.1050.3.6 `param()` [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param () const [inline]
```

Returns the parameter set of the distribution.

Definition at line 4939 of file random.h.

Referenced by `std::operator>>()`.

4.1050.3.7 `param()` [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
 const param_type & __param) [inline]
```

Sets the parameter set of the distribution.

## Parameters

|                      |                                            |
|----------------------|--------------------------------------------|
| <code>__param</code> | The new parameter set of the distribution. |
|----------------------|--------------------------------------------|

Definition at line 4947 of file random.h.

#### 4.1050.3.8 `reset()`

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset () [inline]
```

Resets the distribution state.

Definition at line 4918 of file random.h.

### 4.1050.4 Friends And Related Function Documentation

#### 4.1050.4.1 `operator==`

```
template<typename _RealType = double>
bool operator== (
 const weibull_distribution< _RealType > & __d1,
 const weibull_distribution< _RealType > & __d2) [friend]
```

Return true if two Weibull distributions have the same parameters.

Definition at line 5004 of file random.h.

The documentation for this class was generated from the following files:

- [random.h](#)
- [bits/random.tcc](#)

### 4.1051 `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >` Class Template Reference

#### Public Types

- typedef `basic_string`< char, `char_traits`< char >, `_Byte_alloc` > **byte\_string**
- typedef `wide_string::traits_type::int_type` **int\_type**
- typedef `_Codecvt::state_type` **state\_type**
- typedef `basic_string`< \_Elem, `char_traits`< \_Elem >, `_Wide_alloc` > **wide\_string**



## Public Member Functions

- `wstring_convert` ()
  - `wstring_convert` (\_Codecvt \* \_\_pcvt)
  - `wstring_convert` (\_Codecvt \* \_\_pcvt, state\_type \_\_state)
  - `wstring_convert` (const `byte_string` & \_\_byte\_err, const `wide_string` & \_\_wide\_err=`wide_string`())
  - `wstring_convert` (const `wstring_convert` &)=delete
  - `size_t converted` () const noexcept
  - `wstring_convert` & **operator=** (const `wstring_convert` &)=delete
  - `state_type state` () const
- 
- `wide_string from_bytes` (char \_\_byte)
  - `wide_string from_bytes` (const char \* \_\_ptr)
  - `wide_string from_bytes` (const `byte_string` & \_\_str)
  - `wide_string from_bytes` (const char \* \_\_first, const char \* \_\_last)
- 
- `byte_string to_bytes` (\_Elem \_\_wchar)
  - `byte_string to_bytes` (const \_Elem \* \_\_ptr)
  - `byte_string to_bytes` (const `wide_string` & \_\_wstr)
  - `byte_string to_bytes` (const \_Elem \* \_\_first, const \_Elem \* \_\_last)

## 4.1051.1 Detailed Description

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, typename _Byte_alloc =
allocator<char>>
```

```
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >
```

String conversions.

Definition at line 232 of file `locale_conv.h`.

## 4.1051.2 Constructor &amp; Destructor Documentation

4.1051.2.1 `wstring_convert`() [1/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert () [inline]
```

Default constructor.

Definition at line 241 of file `locale_conv.h`.

**4.1051.2.2 wstring\_convert()** [2/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 _Codecvt * __pcvt) [inline], [explicit]
```

Constructor.

**Parameters**

|                     |                                   |
|---------------------|-----------------------------------|
| <code>__pcvt</code> | The facet to use for conversions. |
|---------------------|-----------------------------------|

Takes ownership of `__pcvt` and will delete it in the destructor.

Definition at line 250 of file `locale_conv.h`.

**4.1051.2.3 `wstring_convert()`** [3/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 _Codecvt * __pcvt,
 state_type __state) [inline]
```

Construct with an initial conversion state.

**Parameters**

|                      |                                   |
|----------------------|-----------------------------------|
| <code>__pcvt</code>  | The facet to use for conversions. |
| <code>__state</code> | Initial conversion state.         |

Takes ownership of `__pcvt` and will delete it in the destructor. The object's conversion state will persist between conversions.

Definition at line 264 of file `locale_conv.h`.

**4.1051.2.4 `wstring_convert()`** [4/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
 const byte_string & __byte_err,
 const wide_string & __wide_err = wide_string()) [inline], [explicit]
```

Construct with error strings.

**Parameters**

|                         |                                                |
|-------------------------|------------------------------------------------|
| <code>__byte_err</code> | A string to return on failed conversions.      |
| <code>__wide_err</code> | A wide string to return on failed conversions. |

Definition at line 277 of file `locale_conv.h`.

### 4.1051.3 Member Function Documentation

#### 4.1051.3.1 converted()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted () const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

Definition at line 367 of file locale\_conv.h.

#### 4.1051.3.2 from\_bytes() [1/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 char __byte) [inline]
```

Convert from bytes.

Definition at line 296 of file locale\_conv.h.

Referenced by std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes().

#### 4.1051.3.3 from\_bytes() [2/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const char * __ptr) [inline]
```

Convert from bytes.

Definition at line 303 of file locale\_conv.h.

References std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes().

## 4.1051.3.4 from\_bytes() [3/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const byte_string & __str) [inline]
```

Convert from bytes.

Definition at line 307 of file locale\_conv.h.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::data(), std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::from\_bytes(), and std::basic\_string< \_CharT, \_Traits, \_Alloc >::size().

## 4.1051.3.5 from\_bytes() [4/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
 const char * __first,
 const char * __last) [inline]
```

Convert from bytes.

Definition at line 314 of file locale\_conv.h.

References std::basic\_string< \_CharT, \_Traits, \_Alloc >::get\_allocator().

## 4.1051.3.6 state()

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state () const
[inline]
```

The final conversion state of the last conversion.

Definition at line 370 of file locale\_conv.h.

## 4.1051.3.7 to\_bytes() [1/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 _Elem __wchar) [inline]
```

Convert to bytes.

Definition at line 330 of file locale\_conv.h.

Referenced by std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >::to\_bytes().

**4.1051.3.8 to\_bytes()** [2/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const _Elem * __ptr) [inline]
```

Convert to bytes.

Definition at line 337 of file locale\_conv.h.

References `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`.

**4.1051.3.9 to\_bytes()** [3/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const wide_string & __wstr) [inline]
```

Convert to bytes.

Definition at line 343 of file locale\_conv.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::data()`, `std::basic_string< _CharT, _Traits, _Alloc >::size()`, and `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes()`.

**4.1051.3.10 to\_bytes()** [4/4]

```
template<typename _Codecvt , typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
 const _Elem * __first,
 const _Elem * __last) [inline]
```

Convert to bytes.

Definition at line 350 of file locale\_conv.h.

References `std::basic_string< _CharT, _Traits, _Alloc >::get_allocator()`.

The documentation for this class was generated from the following file:

- [locale\\_conv.h](#)

## 5 File Documentation

### 5.1 algo.h File Reference

#### Classes

- struct [std::\\_\\_parallel::\\_\\_CRandNumber<\\_MustBeInt>](#)

#### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

#### Functions

- [template<typename \\_RAIter >  
\\_RAIter \*\*std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch\*\* \(\\_RAIter \\_\\_begin, \\_RAIter \\_\\_end, random\\_access\\_iterator\\_tag\)](#)
- [template<typename \\_FIterator, typename \\_IteratorTag >  
\\_FIterator \*\*std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch\*\* \(\\_FIterator \\_\\_begin, \\_FIterator \\_\\_end, \\_IteratorTag\)](#)
- [template<typename \\_FIterator, typename \\_BinaryPredicate, typename \\_IteratorTag >  
\\_FIterator \*\*std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch\*\* \(\\_FIterator \\_\\_begin, \\_FIterator \\_\\_end, \\_BinaryPredicate \\_\\_pred, \\_IteratorTag\)](#)
- [template<typename \\_RAIter, typename \\_BinaryPredicate >  
\\_RAIter \*\*std::\\_\\_parallel::\\_\\_adjacent\\_find\\_switch\*\* \(\\_RAIter \\_\\_begin, \\_RAIter \\_\\_end, \\_BinaryPredicate \\_\\_pred, random\\_access\\_iterator\\_tag\)](#)
- [template<typename \\_RAIter, typename \\_Predicate >  
iterator\\_traits<\\_RAIter>::difference\\_type \*\*std::\\_\\_parallel::\\_\\_count\\_if\\_switch\*\* \(\\_RAIter \\_\\_begin, \\_RAIter \\_\\_end, \\_Predicate \\_\\_pred, random\\_access\\_iterator\\_tag, \[\\\_\\\_gnu\\\_parallel::Parallelism\]\(#\) \\_\\_parallelism\\_tag\)](#)
- [template<typename \\_Iter, typename \\_Predicate, typename \\_IteratorTag >  
iterator\\_traits<\\_Iter>::difference\\_type \*\*std::\\_\\_parallel::\\_\\_count\\_if\\_switch\*\* \(\\_Iter \\_\\_begin, \\_Iter \\_\\_end, \\_Predicate \\_\\_pred, \\_IteratorTag\)](#)
- [template<typename \\_RAIter, typename \\_Tp >  
iterator\\_traits<\\_RAIter>::difference\\_type \*\*std::\\_\\_parallel::\\_\\_count\\_switch\*\* \(\\_RAIter \\_\\_begin, \\_RAIter \\_\\_end, const \\_Tp &\\_\\_value, random\\_access\\_iterator\\_tag, \[\\\_\\\_gnu\\\_parallel::Parallelism\]\(#\) \\_\\_parallelism\\_tag\)](#)
- [template<typename \\_Iter, typename \\_Tp, typename \\_IteratorTag >  
iterator\\_traits<\\_Iter>::difference\\_type \*\*std::\\_\\_parallel::\\_\\_count\\_switch\*\* \(\\_Iter \\_\\_begin, \\_Iter \\_\\_end, const \\_Tp &\\_\\_value, \\_IteratorTag\)](#)
- [template<typename \\_Iter, typename \\_FIterator, typename \\_IteratorTag1, typename \\_IteratorTag2 >  
\\_Iter \*\*std::\\_\\_parallel::\\_\\_find\\_first\\_of\\_switch\*\* \(\\_Iter \\_\\_begin1, \\_Iter \\_\\_end1, \\_FIterator \\_\\_begin2, \\_FIterator \\_\\_end2, \\_IteratorTag1, \\_IteratorTag2\)](#)
- [template<typename \\_RAIter, typename \\_FIterator, typename \\_BinaryPredicate, typename \\_IteratorTag >  
\\_RAIter \*\*std::\\_\\_parallel::\\_\\_find\\_first\\_of\\_switch\*\* \(\\_RAIter \\_\\_begin1, \\_RAIter \\_\\_end1, \\_FIterator \\_\\_begin2, \\_FIterator \\_\\_end2, \\_BinaryPredicate \\_\\_comp, random\\_access\\_iterator\\_tag, \\_IteratorTag\)](#)
- [template<typename \\_Iter, typename \\_FIterator, typename \\_BinaryPredicate, typename \\_IteratorTag1, typename \\_IteratorTag2 >  
\\_Iter \*\*std::\\_\\_parallel::\\_\\_find\\_first\\_of\\_switch\*\* \(\\_Iter \\_\\_begin1, \\_Iter \\_\\_end1, \\_FIterator \\_\\_begin2, \\_FIterator \\_\\_end2, \\_BinaryPredicate \\_\\_comp, \\_IteratorTag1, \\_IteratorTag2\)](#)
- [template<typename \\_Iter, typename \\_Predicate, typename \\_IteratorTag >  
\\_Iter \*\*std::\\_\\_parallel::\\_\\_find\\_if\\_switch\*\* \(\\_Iter \\_\\_begin, \\_Iter \\_\\_end, \\_Predicate \\_\\_pred, \\_IteratorTag\)](#)

- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::find_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_↵`  
`access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IteratorTag >`  
`_Iter std::parallel::find_switch ( _Iter __begin, _Iter __end, const _Tp &__val, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`_RAIter std::parallel::find_switch ( _RAIter __begin, _RAIter __end, const _Tp &__val, random_access_↵`  
`iterator_tag)`
- `template<typename _Iter, typename _Function, typename _IteratorTag >`  
`_Function std::parallel::for_each_switch ( _Iter __begin, _Iter __end, _Function __f, _IteratorTag)`
- `template<typename _RAIter, typename _Function >`  
`_Function std::parallel::for_each_switch ( _RAIter __begin, _RAIter __end, _Function __f, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag >`  
`_OutputIterator std::parallel::generate_n_switch ( _OutputIterator __begin, _Size __n, _Generator __gen,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch ( _RAIter __begin, _Size __n, _Generator __gen, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Generator, typename _IteratorTag >`  
`void std::parallel::generate_switch ( _FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch ( _RAIter __begin, _RAIter __end, _Generator __gen, random_↵`  
`access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::max_element_switch ( _FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _↵`  
`IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::merge_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::parallel::merge_switch ( _Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag,`  
`random_access_iterator_tag)`
- `template<typename _FIterator, typename _Compare, typename _IteratorTag >`  
`_FIterator std::parallel::min_element_switch ( _FIterator __begin, _FIterator __end, _Compare __comp,`  
`_IteratorTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch ( _RAIter __begin, _RAIter __end, _Compare __comp,`  
`random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _IteratorTag >`  
`_FIterator std::parallel::partition_switch ( _FIterator __begin, _FIterator __end, _Predicate __pred, ↵`  
`IteratorTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, random_↵`  
`access_iterator_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_if_switch ( _FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp`  
`&__new_value, _IteratorTag)`



- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch ( _RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp, typename _IteratorTag >`  
`void std::parallel::replace_switch ( _FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch ( _RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_RAIter std::parallel::search_n_switch ( _RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag >`  
`_FIterator std::parallel::search_n_switch ( _FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch ( _FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate >`  
`_RAIter1 std::parallel::search_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_FIterator1 std::parallel::search_switch ( _FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_difference_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_intersection_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_intersection_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_symmetric_difference_switch ( _IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch ( _RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3 >`  
`_OutputIterator std::parallel::set_union_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2 >`  
`_RAIter2 std::parallel::transform1_switch (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3 >`  
`_OutputIterator std::parallel::transform2_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`_OutputIterator std::parallel::unique_copy_switch (_Iter __begin, _Iter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate >`  
`_RandomAccessOutputIterator std::parallel::unique_copy_switch (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _BinaryPredicate >`  
`_FIterator std::parallel::adjacent_find (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits<_Iter>::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & __value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits<_Iter>::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits<_Iter>::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::__parallel::count_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::__parallel::find ( _Iter __begin, _Iter __end, const _Tp &__val)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, _BinaryPredicate __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _FIterator, typename _BinaryPredicate >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2, \_\_gnu\_parallel::sequential\_tag, _BinaryPredicate __comp)`
- `template<typename _Iter, typename _FIterator >`  
`_Iter std::__parallel::find_first_of ( _Iter __begin1, _Iter __end1, _FIterator __begin2, _FIterator __end2)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if ( _Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each ( _Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each ( _Iterator __begin, _Iterator __end, _Function __f)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _FIterator, typename _Generator >`  
`void std::__parallel::generate ( _FIterator __begin, _FIterator __end, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`_OutputIterator std::__parallel::generate_n ( _OutputIterator __begin, _Size __n, _Generator __gen)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element ( _FIterator __begin, _FIterator __end, \_\_gnu\_parallel::Parallelism\_parallelism\_tag)`

- `template<typename _FIterator >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::max_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result, _Compare __comp)`
- `template<typename _IIter1, typename _IIter2, typename _OutputIterator >`  
`_OutputIterator std::__parallel::merge (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↵`  
`_OutputIterator __result)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, __gnu_parallel::Parallelism ↵`  
`__parallelism_tag)`
- `template<typename _FIterator >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp,`  
`__gnu_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Compare >`  
`_FIterator std::__parallel::min_element (_FIterator __begin, _FIterator __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp,`  
`__gnu_parallel::sequential_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`

- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate >`  
`_FIterator std::parallel::partition (_FIterator __begin, _FIterator __end, _Predicate __pred)`
- `template<typename _RAlter >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Tp >`  
`void std::parallel::replace (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _FIterator, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2 >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate >`  
`_FIterator1 std::parallel::search (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _FIterator, typename _Integer, typename _Tp >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)`
- `template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`_FIterator std::parallel::search_n (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`

- [illegible]

- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare, typename _Parallelism >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::default\_parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::parallel\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::multiway\_mergesort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::quicksort\_tag __parallelism)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::balanced\_quicksort\_tag __parallelism)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`\_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, \_OutputIterator __result, \_UnaryOperation __unary_op, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _UnaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter __begin, _Iter __end, _OutputIterator __result, _UnaryOperation __unary_op)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, \_\_gnu\_parallel::\_Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::transform (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out)`
- `template<typename _Iter, typename _OutputIterator, typename _Predicate >`  
`_OutputIterator std::__parallel::unique_copy (_Iter __begin1, _Iter __end1, _OutputIterator __out, _Predicate __pred)`

### 5.1.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algo.h` header.

The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 5.2 `algbase.h` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)



## Functions

- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::parallel::equal_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _↵`  
`Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::equal_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 ↵`  
`__end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`bool std::parallel::lexicographical_compare_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`__begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _R↵`  
`Alter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _RA↵`  
`Iter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _BinaryPredicate`  
`__binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, ↵`  
`BinaryPredicate __binary_pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 ↵`  
`__end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2,`  
`_Iter2 __end2)`

- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`pair< _InputIterator1, _InputIterator2 > std::__parallel::mismatch (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)`

### 5.2.1 Detailed Description

Parallel STL function calls corresponding to the `stl_algobase.h` header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 5.3 algorithm File Reference

### Macros

- `#define _GLIBCXX_ALGORITHM`

### 5.3.1 Detailed Description

This is a Standard C++ Library header.

## 5.4 algorithm File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_ALGORITHM`

## Functions

- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`std::pair< _InputIterator, _OutputIterator > gnu\_cxx::copy\_n ( _InputIterator __first, _Size __count, ↔`  
`OutputIterator __result, std::input\_iterator\_tag)`
- `template<typename _RAIterator, typename _Size, typename _OutputIterator >`  
`std::pair< _RAIterator, _OutputIterator > gnu\_cxx::copy\_n ( _RAIterator __first, _Size __count, ↔`  
`Iterator __result, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int gnu\_cxx::lexicographical\_compare\_3way ( _InputIterator1 __first1, _InputIterator1 __last1, ↔`  
`Iterator2 __first2, _InputIterator2 __last2)`
- `int gnu\_cxx::lexicographical\_compare\_3way (const unsigned char *__first1, const unsigned char *__`  
`last1, const unsigned char *__first2, const unsigned char *__last2)`
- `int gnu\_cxx::lexicographical\_compare\_3way (const char *__first1, const char *__last1, const char *__`  
`_first2, const char *__last2)`
- `template<typename _Tp >`  
`const _Tp & gnu\_cxx::median (const _Tp &__a, const _Tp &__b, const _Tp &__c)`
- `template<typename _Tp, typename _Compare >`  
`const _Tp & gnu\_cxx::median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance >`  
`_RandomAccessIterator gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, ↔`  
`RandomAccessIterator __out, const _Distance __n)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance >`  
`_RandomAccessIterator gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, ↔`  
`RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`std::pair< _InputIterator, _OutputIterator > gnu\_cxx::copy\_n ( _InputIterator __first, _Size __count, ↔`  
`Iterator __result)`
- `template<typename _InputIterator, typename _Tp, typename _Size >`  
`void gnu\_cxx::count ( _InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)`
- `template<typename _InputIterator, typename _Predicate, typename _Size >`  
`void gnu\_cxx::count\_if ( _InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`int gnu\_cxx::lexicographical\_compare\_3way ( _InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2`  
`__first2, _InputIterator2 __last2)`
- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`_RandomAccessIterator gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, ↔`  
`AccessIterator __out_first, _RandomAccessIterator __out_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`_RandomAccessIterator gnu\_cxx::random\_sample ( _InputIterator __first, _InputIterator __last, ↔`  
`AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance >`  
`_OutputIterator gnu\_cxx::random\_sample\_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__out, const _Distance __n)`
- `template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator >`  
`_OutputIterator gnu\_cxx::random\_sample\_n ( _ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__out, const _Distance __n, _RandomNumberGenerator &__rand)`

### 5.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.5 algorithm File Reference

### Macros

- `#define _PARALLEL_ALGORITHM`

### 5.5.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.6 algorithm File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_sample`
- `#define GLIBCXX_EXPERIMENTAL_ALGORITHM`

### Functions

- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>  
<_SampleIterator std::experimental::fundamentals\_v2::sample (_PopulationIterator __first, _PopulationIterator __↵  
__last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)`
- `template<typename _PopulationIterator, typename _SampleIterator, typename _Distance >  
_SampleIterator std::experimental::fundamentals_v2::sample (_PopulationIterator __first, _PopulationIterator  
__last, _SampleIterator __out, _Distance __n)`
- `template<typename _ForwardIterator, typename _Searcher >  
_ForwardIterator std::experimental::fundamentals_v2::search (_ForwardIterator __first, _ForwardIterator __↵  
__last, const _Searcher &__searcher)`
- `template<typename _RandomAccessIterator >  
void std::experimental::fundamentals_v2::shuffle (_RandomAccessIterator __first, _RandomAccessIterator  
__last)`

## 5.6.1 Detailed Description

This is a TS C++ Library header.

## 5.6.2 Function Documentation

## 5.6.2.1 sample()

```
template<typename _PopulationIterator , typename _SampleIterator , typename _Distance , typename
_UniformRandomNumberGenerator >
_SampleIterator std::experimental::fundamentals_v2::sample (
 _PopulationIterator __first,
 _PopulationIterator __last,
 _SampleIterator __out,
 _Distance __n,
 _UniformRandomNumberGenerator && __g)
```

Take a random sample from a population.

Definition at line 61 of file experimental/algorithm.

## 5.7 algorithmfwd.h File Reference

## Namespaces

- [std](#)

## Functions

- `template<typename _Filter >`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter , typename _BinaryPredicate >`  
`constexpr _Filter std::adjacent_find (_Filter, _Filter, _BinaryPredicate)`
- `template<typename _Iter , typename _Predicate >`  
`constexpr bool std::all_of (_Iter, _Iter, _Predicate)`
- `template<typename _Iter , typename _Predicate >`  
`constexpr bool std::any_of (_Iter, _Iter, _Predicate)`
- `template<typename _Filter , typename _Tp >`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter , typename _Tp , typename _Compare >`  
`constexpr bool std::binary_search (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Iter , typename _OIter >`  
`constexpr _OIter std::copy (_Iter, _Iter, _OIter)`
- `template<typename _BIter1 , typename _BIter2 >`  
`constexpr _BIter2 std::copy_backward (_BIter1, _BIter1, _BIter2)`

- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`constexpr _OIter std::copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Iter, typename _Size, typename _OIter >`  
`constexpr _OIter std::copy_n (_Iter, _Size, _OIter)`
- `template<typename _Iter, typename _Tp >`  
`constexpr iterator_traits< _Iter >::difference_type std::count (_Iter, _Iter, const _Tp &)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr iterator_traits< _Iter >::difference_type std::count_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::equal (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr bool std::equal (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _Filter, typename _Tp >`  
`constexpr pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr pair< _Filter, _Filter > std::equal_range (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`constexpr void std::fill (_Filter, _Filter, const _Tp &)`
- `template<typename _OIter, typename _Size, typename _Tp >`  
`constexpr _OIter std::fill_n (_OIter, _Size, const _Tp &)`
- `template<typename _Iter, typename _Tp >`  
`constexpr _Iter std::find (_Iter, _Iter, const _Tp &)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr _Filter1 std::find_end (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr _Filter1 std::find_first_of (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr _Iter std::find_if (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr _Iter std::find_if_not (_Iter, _Iter, _Predicate)`
- `template<typename _Iter, typename _Funct >`  
`constexpr _Funct std::for_each (_Iter, _Iter, _Funct)`
- `template<typename _Filter, typename _Generator >`  
`constexpr void std::generate (_Filter, _Filter, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`constexpr _OIter std::generate_n (_OIter, _Size, _Generator)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`constexpr bool std::includes (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _BIter >`  
`void std::inplace_merge (_BIter, _BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`void std::inplace_merge (_BIter, _BIter, _BIter, _Compare)`
- `template<typename _RAIter >`  
`constexpr bool std::is_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr bool std::is_heap (_RAIter, _RAIter, _Compare)`

- `template<typename _RAIter >`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr _RAIter std::is_heap_until (_RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr bool std::is_partitioned (_Iter, _Iter, _Predicate)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr bool std::is_permutation (_Filter1, _Filter1, _Filter2, _BinaryPredicate)`
- `template<typename _Filter >`  
`constexpr bool std::is_sorted (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr bool std::is_sorted (_Filter, _Filter, _Compare)`
- `template<typename _Filter >`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter std::is_sorted_until (_Filter, _Filter, _Compare)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr void std::iter_swap (_Filter1, _Filter2)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _Compare >`  
`constexpr bool std::lexicographical_compare (_Iter1, _Iter1, _Iter2, _Iter2, _Compare)`
- `template<typename _Filter, typename _Tp >`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &)`
- `template<typename _Filter, typename _Tp, typename _Compare >`  
`constexpr _Filter std::lower_bound (_Filter, _Filter, const _Tp &, _Compare)`
- `template<typename _RAIter >`  
`constexpr void std::make_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::make_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _Tp >`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp std::max (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr _Filter std::max_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter std::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter std::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Tp >`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`

- `template<typename _Tp >`  
`constexpr _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr _Filter std::min_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr _Filter std::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Tp >`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _Filter >`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter)`
- `template<typename _Filter, typename _Compare >`  
`constexpr pair< _Filter, _Filter > std::minmax_element (_Filter, _Filter, _Compare)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2)`
- `template<typename _Iter1, typename _Iter2, typename _BinaryPredicate >`  
`constexpr pair< _Iter1, _Iter2 > std::mismatch (_Iter1, _Iter1, _Iter2, _BinaryPredicate)`
- `template<typename _BIter >`  
`constexpr bool std::next_permutation (_BIter, _BIter)`
- `template<typename _BIter, typename _Compare >`  
`constexpr bool std::next_permutation (_BIter, _BIter, _Compare)`
- `template<typename _Iter, typename _Predicate >`  
`constexpr bool std::none_of (_Iter, _Iter, _Predicate)`
- `template<typename _RAIter >`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::nth_element (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::partial_sort (_RAIter, _RAIter, _RAIter, _Compare)`
- `template<typename _Iter, typename _RAIter >`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter)`
- `template<typename _Iter, typename _RAIter, typename _Compare >`  
`constexpr _RAIter std::partial_sort_copy (_Iter, _Iter, _RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`constexpr _BIter std::partition (_BIter, _BIter, _Predicate)`
- `template<typename _Iter, typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr pair< _Iter1, _Iter2 > std::partition_copy (_Iter, _Iter, _Iter1, _Iter2, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`constexpr _Filter std::partition_point (_Filter, _Filter, _Predicate)`
- `template<typename _RAIter >`  
`constexpr void std::pop_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::pop_heap (_RAIter, _RAIter, _Compare)`



- `template<typename _Blter >`  
`constexpr bool std::prev_permutation (_Blter, _Blter)`
- `template<typename _Blter, typename _Compare >`  
`constexpr bool std::prev_permutation (_Blter, _Blter, _Compare)`
- `template<typename _RAlter >`  
`constexpr void std::push_heap (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Compare >`  
`constexpr void std::push_heap (_RAlter, _RAlter, _Compare)`
- `template<typename _RAlter >`  
`void std::random_shuffle (_RAlter, _RAlter)`
- `template<typename _RAlter, typename _Generator >`  
`void std::random_shuffle (_RAlter, _RAlter, _Generator &&)`
- `template<typename _Filter, typename _Tp >`  
`constexpr _Filter std::remove (_Filter, _Filter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`constexpr _OIter std::remove_copy (_Iter, _Iter, _OIter, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`constexpr _OIter std::remove_copy_if (_Iter, _Iter, _OIter, _Predicate)`
- `template<typename _Filter, typename _Predicate >`  
`constexpr _Filter std::remove_if (_Filter, _Filter, _Predicate)`
- `template<typename _Filter, typename _Tp >`  
`constexpr void std::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Tp >`  
`constexpr _OIter std::replace_copy (_Iter, _Iter, _OIter, const _Tp &, const _Tp &)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp >`  
`constexpr _OIter std::replace_copy_if (_Iter, _Iter, _OIter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`constexpr void std::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Blter >`  
`constexpr void std::reverse (_Blter, _Blter)`
- `template<typename _Blter, typename _OIter >`  
`constexpr _OIter std::reverse_copy (_Blter, _Blter, _OIter)`
- `template<typename _Filter >`  
`constexpr _Filter std::_V2::rotate (_Filter, _Filter, _Filter)`
- `template<typename _Filter, typename _OIter >`  
`constexpr _OIter std::rotate_copy (_Filter, _Filter, _Filter, _OIter)`
- `template<typename _Filter1, typename _Filter2 >`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BinaryPredicate >`  
`constexpr _Filter1 std::search (_Filter1, _Filter1, _Filter2, _Filter2, _BinaryPredicate)`
- `template<typename _Filter, typename _Size, typename _Tp >`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &)`
- `template<typename _Filter, typename _Size, typename _Tp, typename _BinaryPredicate >`  
`constexpr _Filter std::search_n (_Filter, _Filter, _Size, const _Tp &, _BinaryPredicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter std::set_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter std::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`

- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter std::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`constexpr _OIter std::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _RAIter, typename _UGenerator >`  
`void std::shuffle (_RAIter, _RAIter, _UGenerator &&)`
- `template<typename _RAIter >`  
`constexpr void std::sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::sort (_RAIter, _RAIter, _Compare)`
- `template<typename _RAIter >`  
`constexpr void std::sort_heap (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`constexpr void std::sort_heap (_RAIter, _RAIter, _Compare)`
- `template<typename _BIter, typename _Predicate >`  
`_BIter std::stable_partition (_BIter, _BIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::stable_sort (_RAIter, _RAIter)`
- `template<typename _RAIter, typename _Compare >`  
`void std::stable_sort (_RAIter, _RAIter, _Compare)`
- `template<typename _FIter1, typename _FIter2 >`  
`constexpr _FIter2 std::swap_ranges (_FIter1, _FIter1, _FIter2)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`constexpr _OIter std::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BinaryOperation >`  
`constexpr _OIter std::transform (_Iter1, _Iter1, _Iter2, _OIter, _BinaryOperation)`
- `template<typename _FIter >`  
`constexpr _FIter std::unique (_FIter, _FIter)`
- `template<typename _FIter, typename _BinaryPredicate >`  
`constexpr _FIter std::unique (_FIter, _FIter, _BinaryPredicate)`
- `template<typename _Iter, typename _OIter >`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter)`
- `template<typename _Iter, typename _OIter, typename _BinaryPredicate >`  
`constexpr _OIter std::unique_copy (_Iter, _Iter, _OIter, _BinaryPredicate)`
- `template<typename _FIter, typename _Tp >`  
`constexpr _FIter std::upper_bound (_FIter, _FIter, const _Tp &)`
- `template<typename _FIter, typename _Tp, typename _Compare >`  
`constexpr _FIter std::upper_bound (_FIter, _FIter, const _Tp &, _Compare)`

### 5.7.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.8 algorithmfwd.h File Reference

## Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

## Functions

- `template<typename _Filter, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _IterTag)`
- `template<typename _Filter, typename _BiPredicate, typename _IterTag >  
_Filter std::__parallel::__adjacent_find_switch (_Filter, _Filter, _BiPredicate, _IterTag)`
- `template<typename _RAIter, typename _BiPredicate >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)`
- `template<typename _RAIter >  
_RAIter std::__parallel::__adjacent_find_switch (_RAIter __begin, _RAIter __end, random_access_iterator_tag, _tag)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >  
iterator_traits< _Iter >::difference_type std::__parallel::__count_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >  
iterator_traits< _RAIter >::difference_type std::__parallel::__count_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >  
iterator_traits< _Iter >::difference_type std::__parallel::__count_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >  
iterator_traits< _RAIter >::difference_type std::__parallel::__count_switch (_RAIter __begin, _RAIter __end, const _Tp & __value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Filter, typename _IterTag1, typename _IterTag2 >  
_Iter std::__parallel::__find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _Filter, typename _BiPredicate, typename _IterTag >  
_RAIter std::__parallel::__find_first_of_switch (_RAIter, _RAIter, _Filter, _Filter, _BiPredicate, random_access_iterator_tag, _IterTag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >  
_Iter std::__parallel::__find_first_of_switch (_Iter, _Iter, _Filter, _Filter, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _Iter, typename _Predicate, typename _IterTag >  
_Iter std::__parallel::__find_if_switch (_Iter, _Iter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >  
_RAIter std::__parallel::__find_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _RAIter, typename _Tp >  
_RAIter std::__parallel::__find_switch (_RAIter __begin, _RAIter __end, const _Tp & __val, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp, typename _IterTag >  
_Iter std::__parallel::__find_switch (_Iter, _Iter, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Function >  
_Function std::__parallel::__for_each_switch (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Function, typename _IterTag >  
_Function std::__parallel::__for_each_switch (_Iter, _Iter, _Function, _IterTag)`

- `template<typename _OIter, typename _Size, typename _Generator, typename _IterTag >`  
`_OIter std::parallel::generate_n_switch (_OIter, _Size, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Size, typename _Generator >`  
`_RAIter std::parallel::generate_n_switch (_RAIter __begin, _Size __n, _Generator __gen, random_↵`  
`access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Generator, typename _IterTag >`  
`void std::parallel::generate_switch (_Filter, _Filter, _Generator, _IterTag)`
- `template<typename _RAIter, typename _Generator >`  
`void std::parallel::generate_switch (_RAIter __begin, _RAIter __end, _Generator __gen, random_↵`  
`access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`bool std::parallel::lexicographical_compare_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Predicate, _Iter↵`  
`Tag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`bool std::parallel::lexicographical_compare_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`__begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::max_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::max_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, ↵`  
`random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, ↵`  
`typename _IterTag3 >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, _IterTag1, _IterTag2, ↵`  
`_IterTag3)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::parallel::merge_switch (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, random_access_↵`  
`iterator_tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter, typename _Compare, typename _IterTag >`  
`_Filter std::parallel::min_element_switch (_Filter, _Filter, _Compare, _IterTag)`
- `template<typename _RAIter, typename _Compare >`  
`_RAIter std::parallel::min_element_switch (_RAIter __begin, _RAIter __end, _Compare __comp, ↵`  
`random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Predicate >`  
`pair< _RAIter1, _RAIter2 > std::parallel::mismatch_switch (_RAIter1 __begin1, _RAIter1 __end1, _R↵`  
`Alter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`pair< _Iter1, _Iter2 > std::parallel::mismatch_switch (_Iter1, _Iter1, _Iter2, _Predicate, _IterTag1, ↵`  
`IterTag2)`
- `template<typename _Filter, typename _Predicate, typename _IterTag >`  
`_Filter std::parallel::partition_switch (_Filter, _Filter, _Predicate, _IterTag)`
- `template<typename _RAIter, typename _Predicate >`  
`_RAIter std::parallel::partition_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, random_↵`  
`access_iterator_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_if_switch (_Filter, _Filter, _Predicate, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Predicate, typename _Tp >`  
`void std::parallel::replace_if_switch (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &↵`  
`__new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Tp, typename _IterTag >`  
`void std::parallel::replace_switch (_Filter, _Filter, const _Tp &, const _Tp &, _IterTag)`
- `template<typename _RAIter, typename _Tp >`  
`void std::parallel::replace_switch (_RAIter __begin, _RAIter __end, const _Tp & __old_value, const _Tp ↵`  
`& __new_value, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism_tag)`

- `template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_RAIter std::parallel::search_n_switch (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_↵`  
`_access_iterator_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag >`  
`_Filter std::parallel::search_n_switch (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, _IterTag)`
- `template<typename _Filter1, typename _Filter2, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2, typename _BiPredicate >`  
`_RAIter1 std::parallel::search_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_↵`  
`access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2 >`  
`_Filter1 std::parallel::search_switch (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter1, typename _RAIter2 >`  
`_RAIter1 std::parallel::search_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2`  
`__end2, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_difference_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`__begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_↵`  
`_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2 ,`  
`typename _IterTag3 >`  
`_Olter std::parallel::set_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1,`  
`_IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_intersection_switch (_RAIter1 __begin1, _RAIter1 __end1, _RA↵`  
`Iter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag,`  
`random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2 ,`  
`typename _IterTag3 >`  
`_Olter std::parallel::set_intersection_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _Iter↵`  
`Tag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_symmetric_difference_switch (_RAIter1 __begin1, _RAIter1 __end1,`  
`_RAIter2 __begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_↵`  
`tag, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2 ,`  
`typename _IterTag3 >`  
`_Olter std::parallel::set_symmetric_difference_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, ↵`  
`Predicate, _IterTag1, _IterTag2, _IterTag3)`
- `template<typename _RAIter1, typename _RAIter2, typename _Output_RAlter, typename _Predicate >`  
`_Output_RAlter std::parallel::set_union_switch (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↵`  
`begin2, _RAIter2 __end2, _Output_RAlter __result, _Predicate __pred, random_access_iterator_tag, random_↵`  
`_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate, typename _Olter, typename _IterTag1, typename _IterTag2 ,`  
`typename _IterTag3 >`  
`_Olter std::parallel::set_union_switch (_Iter1, _Iter1, _Iter2, _Iter2, _Olter, _Predicate, _IterTag1, ↵`  
`IterTag2, _IterTag3)`
- `template<typename _Iter, typename _Olter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2 >`  
`_Olter std::parallel::transform1_switch (_Iter, _Iter, _Olter, _UnaryOperation, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RAOlter, typename _UnaryOperation >`  
`_RAOlter std::parallel::transform1_switch (_RAIter, _RAIter, _RAOlter, _UnaryOperation, random_↵`  
`access_iterator_tag, random_access_iterator_tag, gnu\_parallel::Parallelism __parallelism= gnu\_parallel::parallel\_balanced)`

- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation >`  
`_RAIter3 std::parallel::transform2_switch (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation,`  
`random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism`  
`__parallelism=\_\_gnu\_parallel::parallel\_balanced)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename`  
`_Tag3 >`  
`_OIter std::parallel::transform2_switch (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, _Tag1, _Tag2, ↵`  
`Tag3)`
- `template<typename _Iter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2 >`  
`_OIter std::parallel::unique_copy_switch (_Iter, _Iter, _OIter, _Predicate, _IterTag1, _IterTag2)`
- `template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate >`  
`_RandomAccess_OIter std::parallel::unique_copy_switch (_RAIter, _RAIter, _RandomAccess_OIter, ↵`  
`Predicate, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate)`
- `template<typename _Filter, typename _BiPredicate >`  
`_Filter std::parallel::adjacent_find (_Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & ↵`  
`value, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & ↵`  
`value, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp >`  
`iterator_traits< _Iter >::difference_type std::parallel::count (_Iter __begin, _Iter __end, const _Tp & ↵`  
`value)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate ↵`  
`pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate ↵`  
`pred, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Predicate >`  
`iterator_traits< _Iter >::difference_type std::parallel::count_if (_Iter __begin, _Iter __end, _Predicate ↵`  
`pred)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred,`  
`\_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool std::parallel::equal (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Predicate __pred)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp & __val, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Iter std::parallel::find (_Iter __begin, _Iter __end, const _Tp & __val)`

- `template<typename _Iter, typename _Filter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Filter, typename _BiPredicate >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter, _BiPredicate)`
- `template<typename _Iter, typename _Filter >`  
`_Iter std::__parallel::find_first_of (_Iter, _Iter, _Filter, _Filter)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Predicate >`  
`_Iter std::__parallel::find_if (_Iter __begin, _Iter __end, _Predicate __pred)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter __begin, _Iter __end, _Function __f, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iterator, typename _Function >`  
`_Function std::__parallel::for_each (_Iterator __begin, _Iterator __end, _Function __f, \_\_gnu\_parallel::Parallelism  
\_\_parallelism\_tag)`
- `template<typename _Iter, typename _Function >`  
`_Function std::__parallel::for_each (_Iter, _Iter, _Function)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Generator >`  
`void std::__parallel::generate (_Filter, _Filter, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _OIter, typename _Size, typename _Generator >`  
`_OIter std::__parallel::generate_n (_OIter, _Size, _Generator, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`constexpr bool std::__parallel::lexicographical_compare (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, _Iter2 __end2, _Predicate __pred)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::max_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Compare >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Compare)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::merge (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, \_\_gnu\_parallel::Parallelism __parallelism_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Compare >`  
`_Filter std::__parallel::min_element (_Filter, _Filter, _Compare, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2 >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2)`
- `template<typename _Iter1, typename _Iter2, typename _Predicate >`  
`pair< _Iter1, _Iter2 > std::__parallel::mismatch (_Iter1 __begin1, _Iter1 __end1, _Iter2 __begin2, \_\_gnu\_parallel::sequential\_tag, _Predicate __pred)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end)`
- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`



- `template<typename _Filter, typename _Predicate >`  
`_Filter std::__parallel::partition (_Filter, _Filter, _Predicate)`
- `template<typename _RAlter >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &__rand, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAlter >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end)`
- `template<typename _RAlter, typename _RandomNumberGenerator >`  
`void std::__parallel::random_shuffle (_RAlter __begin, _RAlter __end, _RandomNumberGenerator &&__rand)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Tp >`  
`void std::__parallel::replace (_Filter, _Filter, const _Tp &, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Predicate, typename _Tp >`  
`void std::__parallel::replace_if (_Filter, _Filter, _Predicate, const _Tp &, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2 >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter1, typename _Filter2, typename _BiPredicate >`  
`_Filter1 std::__parallel::search (_Filter1, _Filter1, _Filter2, _Filter2, _BiPredicate)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Filter, typename _Integer, typename _Tp >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &)`
- `template<typename _Filter, typename _Integer, typename _Tp, typename _BiPredicate >`  
`_Filter std::__parallel::search_n (_Filter, _Filter, _Integer, const _Tp &, _BiPredicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter >`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate >`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter >`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate >`  
`_Olter std::__parallel::set_difference (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter >`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter, typename _Predicate >`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Ilter1, typename _Ilter2, typename _Olter >`  
`_Olter std::__parallel::set_intersection (_Ilter1, _Ilter1, _Ilter2, _Ilter2, _Olter)`

- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_intersection (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_symmetric_difference (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::set_union (_Iter1, _Iter1, _Iter2, _Iter2, _OIter, _Predicate)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIter >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end)`
- `template<typename _RAIter, typename _Compare >`  
`void std::__parallel::stable_sort (_RAIter __begin, _RAIter __end, _Compare __comp)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _UnaryOperation >`  
`_OIter std::__parallel::transform (_Iter, _Iter, _OIter, _UnaryOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _OIter, typename _BiOperation >`  
`_OIter std::__parallel::transform (_Iter1, _Iter1, _Iter2, _OIter, _BiOperation, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter, typename _Predicate >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >`  
`_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter)`

- `template<typename _Iter, typename _OIter, typename _Predicate >  
_OIter std::__parallel::unique_copy (_Iter, _Iter, _OIter, _Predicate)`

### 5.8.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 5.9 aligned\_buffer.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.9.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.10 alloc\_traits.h File Reference

### Classes

- struct [std::allocator\\_traits< \\_Alloc >](#)
- struct [std::allocator\\_traits< allocator< \\_Tp > >](#)

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_allocator_traits_is_always_equal`

### Typedefs

- `template<typename _Alloc, typename _Up >  
using std::__alloc_rebind = typename __allocator_traits_base::template __rebind< _Alloc, _Up >::type`
- `template<typename _Alloc >  
using std::__RequireAllocator = typename enable_if< __is_allocator< _Alloc >::value, _Alloc >::type`
- `template<typename _Alloc >  
using std::__RequireNotAllocator = typename enable_if<!__is_allocator< _Alloc >::value, _Alloc >::type`

## Functions

- `template<typename _Alloc >`  
`constexpr void std::__alloc_on_copy (_Alloc &__one, const _Alloc &__two)`
- `template<typename _Alloc >`  
`constexpr _Alloc std::__alloc_on_copy (const _Alloc &__a)`
- `template<typename _Alloc >`  
`constexpr void std::__alloc_on_move (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`  
`constexpr void std::__alloc_on_swap (_Alloc &__one, _Alloc &__two)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_copy (_Alloc &__one, const _Alloc &__two, true_type)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_copy (_Alloc &, const _Alloc &, false_type)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_move (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_move (_Alloc &, _Alloc &, false_type)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_swap (_Alloc &__one, _Alloc &__two, true_type)`
- `template<typename _Alloc >`  
`void std::__do_alloc_on_swap (_Alloc &, _Alloc &, false_type)`
- `template<typename _ForwardIterator, typename _Allocator >`  
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, _Allocator &__alloc)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last, allocator< _Tp > &)`

### 5.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.11 alloc\_traits.h File Reference

### Classes

- struct [`\_\_gnu\_cxx::\_\_alloc\_traits<\_Alloc, typename >`](#)

### Namespaces

- [`\_\_gnu\_cxx`](#)

### 5.11.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.12 `allocated_ptr.h` File Reference

### Classes

- struct `std::__allocated_ptr<_Alloc>`

### Namespaces

- `std`

### Functions

- template<typename `_Alloc`>  
`__allocated_ptr<_Alloc>` `std::__allocate_guarded` (`_Alloc` &`__a`)

#### 5.12.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.13 `allocator.h` File Reference

### Classes

- class `std::allocator<_Tp>`
- class `std::allocator<void>`

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_incomplete_container_elements`

### Functions

- template<typename `_T1`, typename `_T2`>  
constexpr bool **`std::operator!=`** (const allocator<`_T1`> &, const allocator<`_T2`> &) noexcept
- template<typename `_T1`, typename `_T2`>  
constexpr bool **`std::operator==`** (const allocator<`_T1`> &, const allocator<`_T2`> &) noexcept

### 5.13.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.14 any File Reference

### Macros

- `#define _GLIBCXX_ANY`

### 5.14.1 Detailed Description

This is a Standard C++ Library header.

## 5.15 any File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::any](#)
- class [std::experimental::fundamentals\\_v1::bad\\_any\\_cast](#)

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_any`
- `#define _GLIBCXX_EXPERIMENTAL_ANY`

### Functions

- `template<typename _ValueType >`  
`_ValueType std::experimental::fundamentals_v1::any_cast (const any &__any)`
- `void std::experimental::fundamentals_v1::swap (any &__x, any &__y) noexcept`
  
- `template<typename _ValueType >`  
`_ValueType std::experimental::fundamentals_v1::any_cast (any &__any)`
- `template<typename _ValueType , typename enable_if<is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>`  
`_ValueType std::experimental::fundamentals_v1::any_cast (any &&__any)`
  
- `template<typename _ValueType >`  
`const _ValueType * std::experimental::fundamentals_v1::any_cast (const any * __any) noexcept`
- `template<typename _ValueType >`  
`_ValueType * std::experimental::fundamentals_v1::any_cast (any * __any) noexcept`

## 5.15.1 Detailed Description

This is a TS C++ Library header.

## 5.16 array File Reference

## Classes

- struct [std::array<\\_Tp, \\_Nm >](#)
- struct [std::tuple\\_element<\\_Int, \\_Tp >](#)
- struct [std::tuple\\_element<\\_Int, ::array<\\_Tp, \\_Nm > >](#)
- struct [std::tuple\\_size<\\_Tp >](#)
- struct [std::tuple\\_size<::array<\\_Tp, \\_Nm > >](#)

## Namespaces

- [std](#)

## Macros

- `#define \_GLIBCXX\_ARRAY`

## Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp & std::get (array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr _Tp && std::get (array< _Tp, _Nm > &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp & std::get (const array< _Tp, _Nm > &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>`  
`constexpr const _Tp && std::get (const array< _Tp, _Nm > &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator!= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator< (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator<= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator== (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator> (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr bool std::operator>= (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)`
- `template<typename _Tp, std::size_t _Nm>`  
`constexpr enable_if< ::__array_traits< _Tp, _Nm >::is_swappable::value >::type std::swap (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))`
- `template<typename _Tp, std::size_t _Nm>`  
`enable_if< !::__array_traits< _Tp, _Nm >::is_swappable::value >::type std::swap (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete`

### 5.16.1 Detailed Description

This is a Standard C++ Library header.

## 5.17 array File Reference

### Classes

- struct [std::tuple\\_element<\\_Int, std::\\_\\_debug::array<\\_Tp, \\_Nm>>](#)
- struct [std::tuple\\_size<std::\\_\\_debug::array<\\_Tp, \\_Nm>>](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define \_GLIBCXX\_DEBUG\_ARRAY`

### Functions

- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp & std::\_\_debug::get (array<_Tp, _Nm> &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr _Tp && std::\_\_debug::get (array<_Tp, _Nm> &&__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp & std::\_\_debug::get (const array<_Tp, _Nm> &__arr) noexcept`
- `template<std::size_t _Int, typename _Tp, std::size_t _Nm>  
constexpr const _Tp && std::\_\_debug::get (const array<_Tp, _Nm> &&__arr) noexcept`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator!= (const array<_Tp, _Nm> &__one, const array<_Tp, _Nm> &__two)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator< (const array<_Tp, _Nm> &__a, const array<_Tp, _Nm> &__b)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator<= (const array<_Tp, _Nm> &__one, const array<_Tp, _Nm> &__two)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator== (const array<_Tp, _Nm> &__one, const array<_Tp, _Nm> &__two)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator> (const array<_Tp, _Nm> &__one, const array<_Tp, _Nm> &__two)`
- `template<typename _Tp, std::size_t _Nm>  
constexpr bool std::\_\_debug::operator>= (const array<_Tp, _Nm> &__one, const array<_Tp, _Nm> &__two)`
- `template<typename _Tp, size_t _Nm>  
enable_if<!::__array_traits<_Tp, _Nm>::is_swappable::value>::type std::\_\_debug::swap (array<_Tp, _Nm> &, array<_Tp, _Nm> &)=delete`
- `template<typename _Tp, std::size_t _Nm>  
constexpr void std::\_\_debug::swap (array<_Tp, _Nm> &__one, array<_Tp, _Nm> &__two)  
noexcept(noexcept(__one.swap(__two)))`



## 5.17.1 Detailed Description

This is a Standard C++ Library header.

## 5.18 array File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_make_array`
- `#define _GLIBCXX_EXPERIMENTAL_ARRAY`

## Functions

- `template<typename _Tp, size_t _Nm, size_t... _Idx>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals\_v2::\_\_to\_array (_Tp(&_↵  
_a)[_Nm], index_sequence< _Idx... >)`
- `template<typename _Dest = void, typename... _Types>`  
`constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::fundamentals\_v2::  
(_Types &&... __t)`
- `template<typename _Tp, size_t _Nm>`  
`constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals\_v2::to\_array (_Tp(&__a)[_Nm])`  
`noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)`

## 5.18.1 Detailed Description

This is a TS C++ Library header.

## 5.19 assertions.h File Reference

## Macros

- `#define __glibcxx_requires_non_empty_range(_First, _Last)`
- `#define __glibcxx_requires_nonempty()`
- `#define __glibcxx_requires_subscript(_N)`
- `#define _GLIBCXX_DEBUG_ASSERT(_Condition)`
- `#define _GLIBCXX_DEBUG_ONLY(_Statement)`
- `#define _GLIBCXX_DEBUG_PEDASSERT(_Condition)`

### 5.19.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.20 `assoc_container.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::basic\\_branch](#)< Key, Mapped, Tag, Node\_Update, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::basic\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Resize\_Policy, Store\_Hash, Tag, Policy\_Tl, \_Alloc >
- class [\\_\\_gnu\\_pbds::cc\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::gp\\_hash\\_table](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >
- class [\\_\\_gnu\\_pbds::list\\_update](#)< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >
- class [\\_\\_gnu\\_pbds::tree](#)< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie](#)< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BRANCH_BASE`
- `#define PB_DS_CC_HASH_BASE`
- `#define PB_DS_GP_HASH_BASE`
- `#define PB_DS_HASH_BASE`
- `#define PB_DS_LU_BASE`
- `#define PB_DS_TREE_BASE`
- `#define PB_DS_TREE_NODE_AND_IT_TRAITS`
- `#define PB_DS_TRIE_BASE`
- `#define PB_DS_TRIE_NODE_AND_IT_TRAITS`

### 5.20.1 Detailed Description

Contains associative containers.

## 5.21 atomic File Reference

## Classes

- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic< \\_Tp \\* >](#)
- struct [std::atomic< bool >](#)
- struct [std::atomic< char >](#)
- struct [std::atomic< char16\\_t >](#)
- struct [std::atomic< char32\\_t >](#)
- struct [std::atomic< int >](#)
- struct [std::atomic< long >](#)
- struct [std::atomic< long long >](#)
- struct [std::atomic< short >](#)
- struct [std::atomic< signed char >](#)
- struct [std::atomic< unsigned char >](#)
- struct [std::atomic< unsigned int >](#)
- struct [std::atomic< unsigned long >](#)
- struct [std::atomic< unsigned long long >](#)
- struct [std::atomic< unsigned short >](#)
- struct [std::atomic< wchar\\_t >](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX20_INIT(I)`
- `#define _GLIBCXX_ATOMIC`

## Typedefs

- `template<typename _Tp >`  
using [std::\\_\\_atomic\\_diff\\_t](#) = typename atomic< \_Tp >::difference\_type
- `template<typename _Tp >`  
using [std::\\_\\_atomic\\_val\\_t](#) = typename atomic< \_Tp >::value\_type
- `typedef atomic< bool > std::atomic\_bool`
- `typedef atomic< char > std::atomic\_char`
- `typedef atomic< char16_t > std::atomic\_char16\_t`
- `typedef atomic< char32_t > std::atomic\_char32\_t`
- `typedef atomic< int > std::atomic\_int`
- `typedef atomic< int16_t > std::atomic\_int16\_t`
- `typedef atomic< int32_t > std::atomic\_int32\_t`
- `typedef atomic< int64_t > std::atomic\_int64\_t`
- `typedef atomic< int8_t > std::atomic\_int8\_t`
- `typedef atomic< int_fast16_t > std::atomic\_int\_fast16\_t`

- `typedef atomic< int_fast32_t > std::atomic_int_fast32_t`
- `typedef atomic< int_fast64_t > std::atomic_int_fast64_t`
- `typedef atomic< int_fast8_t > std::atomic_int_fast8_t`
- `typedef atomic< int_least16_t > std::atomic_int_least16_t`
- `typedef atomic< int_least32_t > std::atomic_int_least32_t`
- `typedef atomic< int_least64_t > std::atomic_int_least64_t`
- `typedef atomic< int_least8_t > std::atomic_int_least8_t`
- `typedef atomic< intmax_t > std::atomic_intmax_t`
- `typedef atomic< intptr_t > std::atomic_intptr_t`
- `typedef atomic< long long > std::atomic_llong`
- `typedef atomic< long > std::atomic_long`
- `typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t`
- `typedef atomic< signed char > std::atomic_schar`
- `typedef atomic< short > std::atomic_short`
- `typedef atomic< size_t > std::atomic_size_t`
- `typedef atomic< unsigned char > std::atomic_uchar`
- `typedef atomic< unsigned int > std::atomic_uint`
- `typedef atomic< uint16_t > std::atomic_uint16_t`
- `typedef atomic< uint32_t > std::atomic_uint32_t`
- `typedef atomic< uint64_t > std::atomic_uint64_t`
- `typedef atomic< uint8_t > std::atomic_uint8_t`
- `typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t`
- `typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t`
- `typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t`
- `typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t`
- `typedef atomic< uint_least16_t > std::atomic_uint_least16_t`
- `typedef atomic< uint_least32_t > std::atomic_uint_least32_t`
- `typedef atomic< uint_least64_t > std::atomic_uint_least64_t`
- `typedef atomic< uint_least8_t > std::atomic_uint_least8_t`
- `typedef atomic< uintmax_t > std::atomic_uintmax_t`
- `typedef atomic< uintptr_t > std::atomic_uintptr_t`
- `typedef atomic< unsigned long long > std::atomic_ullong`
- `typedef atomic< unsigned long > std::atomic_ulong`
- `typedef atomic< unsigned short > std::atomic_ushort`
- `typedef atomic< wchar_t > std::atomic_wchar_t`

## Functions

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_strong_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`

- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_compare_exchange_weak_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_exchange_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_add_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_and_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_or_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`

- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)`  
`noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_sub_explicit (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_fetch_xor_explicit (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `void std::atomic_flag_clear (atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear (volatile atomic_flag *__a) noexcept`
- `void std::atomic_flag_clear_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `void std::atomic_flag_clear_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set (atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set (volatile atomic_flag *__a) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (atomic_flag *__a, memory_order __m) noexcept`
- `bool std::atomic_flag_test_and_set_explicit (volatile atomic_flag *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_init (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`bool std::atomic_is_lock_free (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load (const volatile atomic< _ITp > *__a) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`_ITp std::atomic_load_explicit (const volatile atomic< _ITp > *__a, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`
- `template<typename _ITp >`  
`void std::atomic_store_explicit (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept`

## 5.21.1 Detailed Description

This is a Standard C++ Library header.

## 5.22 atomic\_base.h File Reference

## Classes

- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_ITp >](#)
- struct [std::\\_\\_atomic\\_base< \\_PTp \\* >](#)
- struct [std::\\_\\_atomic\\_flag\\_base](#)
- struct [std::atomic< \\_Tp >](#)
- struct [std::atomic\\_flag](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX20_INIT(l)`
- `#define _GLIBCXX_ALWAYS_INLINE`
- `#define ATOMIC_FLAG_INIT`
- `#define ATOMIC_VAR_INIT(_VI)`

## Typedefs

- typedef unsigned char [std::\\_\\_atomic\\_flag\\_data\\_type](#)
- typedef enum [std::memory\\_order](#) [std::memory\\_order](#)

## Enumerations

- enum [\\_\\_memory\\_order\\_modifier](#) { [\\_\\_memory\\_order\\_mask](#), [\\_\\_memory\\_order\\_modifier\\_mask](#), [\\_\\_memory\\_order\\_hle\\_acquire](#), [\\_\\_memory\\_order\\_hle\\_release](#) }
- enum [std::memory\\_order](#) { [memory\\_order\\_relaxed](#), [memory\\_order\\_consume](#), [memory\\_order\\_acquire](#), [memory\\_order\\_release](#), [memory\\_order\\_acq\\_rel](#), [memory\\_order\\_seq\\_cst](#) }

## Functions

- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order](#) (memory\_order \_\_m) noexcept
- constexpr memory\_order [std::\\_\\_cmpexch\\_failure\\_order2](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_signal\\_fence](#) (memory\_order \_\_m) noexcept
- void [std::atomic\\_thread\\_fence](#) (memory\_order \_\_m) noexcept
- template<typename \_Tp >  
\_Tp [std::kill\\_dependency](#) (\_Tp \_\_y) noexcept
- constexpr memory\_order [std::operator&](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)
- constexpr memory\_order [std::operator|](#) (memory\_order \_\_m, \_\_memory\_order\_modifier \_\_mod)

### 5.22.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 5.23 `atomic_futex.h` File Reference

### Namespaces

- [std](#)

### 5.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.24 `atomic_lockfree_defines.h` File Reference

### Macros

- #define [ATOMIC\\_BOOL\\_LOCK\\_FREE](#)
- #define **ATOMIC\_CHAR16\_T\_LOCK\_FREE**
- #define **ATOMIC\_CHAR32\_T\_LOCK\_FREE**
- #define **ATOMIC\_CHAR\_LOCK\_FREE**
- #define **ATOMIC\_INT\_LOCK\_FREE**
- #define **ATOMIC\_LLONG\_LOCK\_FREE**
- #define **ATOMIC\_LONG\_LOCK\_FREE**
- #define **ATOMIC\_POINTER\_LOCK\_FREE**
- #define **ATOMIC\_SHORT\_LOCK\_FREE**
- #define **ATOMIC\_WCHAR\_T\_LOCK\_FREE**

### 5.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<atomic>`.

## 5.25 `atomic_word.h` File Reference

### Macros

- #define **\_GLIBCXX\_READ\_MEM\_BARRIER**
- #define **\_GLIBCXX\_WRITE\_MEM\_BARRIER**



## Typedefs

- typedef int **\_Atomic\_word**

## 5.25.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.26 atomicity.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- #define **\_GLIBCXX\_READ\_MEM\_BARRIER**
- #define **\_GLIBCXX\_WRITE\_MEM\_BARRIER**

## Functions

- void **\_\_gnu\_cxx::\_\_atomic\_add** (volatile \_Atomic\_word \*, int) noexcept
- void **\_\_gnu\_cxx::\_\_atomic\_add\_dispatch** (\_Atomic\_word \*\_\_mem, int \_\_val)
- void **\_\_gnu\_cxx::\_\_atomic\_add\_single** (\_Atomic\_word \*\_\_mem, int \_\_val)
- \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add** (volatile \_Atomic\_word \*, int) noexcept
- \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add\_dispatch** (\_Atomic\_word \*\_\_mem, int \_\_val)
- \_Atomic\_word **\_\_gnu\_cxx::\_\_exchange\_and\_add\_single** (\_Atomic\_word \*\_\_mem, int \_\_val)

## 5.26.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.27 auto\_ptr.h File Reference

## Classes

- class [std::auto\\_ptr<\\_Tp>](#)
- struct [std::auto\\_ptr\\_ref<\\_Tp1>](#)

## Namespaces

- [std](#)

### 5.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.28 `backward_warning.h` File Reference

### 5.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.29 `balanced_quicksort.h` File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal<\\_RAIter >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_sort\\_qsb](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_conquer](#) (\_QSBThreadLocal<\_RAIter > \*\_\_tls, \_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_iam, \_ThreadIndex \_\_num\_threads, bool \_\_parent\_wait)
- template<typename \_RAIter, typename \_Compare >  
[std::iterator\\_traits<\\_RAIter >::difference\\_type](#) [\\_\\_gnu\\_parallel::\\_\\_qsb\\_divide](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_Compare \_\_comp, \_ThreadIndex \_\_num\_threads)
- template<typename \_RAIter, typename \_Compare >  
void [\\_\\_gnu\\_parallel::\\_\\_qsb\\_local\\_sort\\_with\\_helping](#) (\_QSBThreadLocal<\_RAIter > \*\_\_tls, \_Compare &\_\_comp, \_ThreadIndex \_\_iam, bool \_\_wait)

### 5.29.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.

It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in

P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.

This file is a GNU parallel extension to the Standard C++ Library.

## 5.30 base.h File Reference

## Classes

- class [\\_\\_gnu\\_parallel::\\_\\_binder1st<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_binder2nd<\\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate<\\_Predicate, argument\\_type>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_EqualFromLess<\\_T1, \\_T2, \\_Compare>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_EqualTo<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Less<\\_T1, \\_T2>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Multiplies<\\_Tp1, \\_Tp2, \\_Result>](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_Plus<\\_Tp1, \\_Tp2, \\_Result>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequence<\\_Tp, \\_DifferenceTp>](#)
- class [\\_\\_gnu\\_parallel::\\_\\_PseudoSequenceIterator<\\_Tp, \\_DifferenceTp>](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)
- [\\_\\_gnu\\_sequential](#)
- [std](#)
- [std::\\_\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_PARALLEL\_ASSERT(_Condition)`

## Functions

- void [\\_\\_gnu\\_parallel::\\_\\_decode2](#) ([\\_CASable](#) \_\_x, int &\_\_a, int &\_\_b)
- [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_\\_encode2](#) (int \_\_a, int \_\_b)
- [\\_ThreadIndex](#) [\\_\\_gnu\\_parallel::\\_\\_get\\_max\\_threads](#) ()
- bool [\\_\\_gnu\\_parallel::\\_\\_is\\_parallel](#) (const [\\_Parallelism](#) \_\_p)
- template<typename [\\_RAlter](#) , typename [\\_Compare](#) >  
[\\_RAlter](#) [\\_\\_gnu\\_parallel::\\_\\_median\\_of\\_three\\_iterators](#) ([\\_RAlter](#) \_\_a, [\\_RAlter](#) \_\_b, [\\_RAlter](#) \_\_c, [\\_Compare](#) \_\_comp)
- template<typename [\\_Size](#) >  
[\\_Size](#) [\\_\\_gnu\\_parallel::\\_\\_rd\\_log2](#) ([\\_Size](#) \_\_n)
- template<typename [\\_Tp](#) >  
const [\\_Tp](#) & [\\_\\_gnu\\_parallel::\\_\\_max](#) (const [\\_Tp](#) &\_\_a, const [\\_Tp](#) &\_\_b)
- template<typename [\\_Tp](#) >  
const [\\_Tp](#) & [\\_\\_gnu\\_parallel::\\_\\_min](#) (const [\\_Tp](#) &\_\_a, const [\\_Tp](#) &\_\_b)

## 5.30.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

## 5.31 `basic_file.h` File Reference

### Namespaces

- [std](#)

### 5.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 5.32 `basic_ios.h` File Reference

### Classes

- class [std::basic\\_ios<\\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Functions

- `template<typename _Facet>  
const _Facet & std::__check_facet (const _Facet * __f)`

### 5.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

## 5.33 `basic_ios.tcc` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _BASIC_IOS_TCC`

## 5.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.34 `basic_iterator.h` File Reference

## 5.34.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

5.35 `basic_string.h` File Reference

## Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- struct `std::hash<string>`
- struct `std::hash<u16string>`
- struct `std::hash<u32string>`
- struct `std::hash<wstring>`

## Namespaces

- `std`

## Macros

- `#define __cpp_lib_string_udls`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream<_CharT, _Traits> &std::getline(basic_istream<_CharT, _Traits> &__is, basic_string<_CharT, _Traits, _Alloc> &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream<_CharT, _Traits> &std::getline(basic_istream<_CharT, _Traits> &__is, basic_string<_CharT, _Traits, _Alloc> &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream<_CharT, _Traits> &std::getline(basic_istream<_CharT, _Traits> &&__is, basic_string<_CharT, _Traits, _Alloc> &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc>`  
`basic_istream<_CharT, _Traits> &std::getline(basic_istream<_CharT, _Traits> &&__is, basic_string<_CharT, _Traits, _Alloc> &__str)`

- `template<>`  
`basic_istream< char > & std::getline (basic_istream< char > &__in, basic_string< char > &__str, char __↵`  
`delim)`
- `template<>`  
`basic_istream< wchar_t > & std::getline (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str,`  
`wchar_t __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator!= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char > std::literals::string_literals::operator""s (const char`  
`*__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< wchar_t > std::literals::string_literals::operator""s (const`  
`wchar_t *__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char16_t > std::literals::string_literals::operator""s (const`  
`char16_t *__str, size_t __len)`
- `_GLIBCXX_DEFAULT_ABI_TAG basic_string< char32_t > std::literals::string_literals::operator""s (const`  
`char32_t *__str, size_t __len)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _↵`  
`_Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc`  
`> &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`_CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const`  
`basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs,`  
`basic_string< _CharT, _Traits, _Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, basic_string< _CharT, _Traits,`  
`_Alloc > &&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc >`  
`&&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const`  
`_CharT *__rhs)`

- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_string< _CharT, _Traits, _Alloc > std::operator+ (basic_string< _CharT, _Traits, _Alloc > &&__lhs, ↵`  
`CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic↵`  
`_string< _CharT, _Traits, _Alloc > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, bool >::__type std::operator== (const basic_string<`  
`_CharT > &__lhs, const basic_string< _CharT > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator== (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits,`  
`_Alloc > &__rhs) noexcept`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`bool std::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string<`  
`_CharT, _Traits, _Alloc > &__str)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > &__is, basic_string< char > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs)`  
`noexcept(/*conditional */)`
- `string std::to_string (int __val)`

- string **std::to\_string** (unsigned \_\_val)
- string **std::to\_string** (long \_\_val)
- string **std::to\_string** (unsigned long \_\_val)
- string **std::to\_string** (long long \_\_val)
- string **std::to\_string** (unsigned long long \_\_val)

#### 5.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

### 5.36 `basic_string.tcc` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _BASIC_STRING_TCC`

#### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc >  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)`

#### 5.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

### 5.37 `bin_search_tree.hpp` File Reference

#### Namespaces

- [\\_\\_gnu\\_pbds](#)



## Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node)`
- `#define PB_DS_BIN_TREE_NAME`
- `#define PB_DS_BIN_TREE_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

## 5.37.1 Detailed Description

Contains an implementation class for binary search tree.

5.38 `binary_heap.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_ENTRY_CMP_DEC`
- `#define PB_DS_RESIZE_POLICY_DEC`

## 5.38.1 Detailed Description

Contains an implementation class for a binary heap.

5.39 `binders.h` File Reference

## Classes

- class [std::binder1st< \\_Operation >](#)
- class [std::binder2nd< \\_Operation >](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _Operation , typename _Tp >`  
`binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)`
- `template<typename _Operation , typename _Tp >`  
`binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)`

### 5.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.40 binomial\_heap\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.40.1 Detailed Description

Contains an implementation class for a binomial heap.

## 5.41 binomial\_heap\_base\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_ASSERT_VALID_COND(X, _StrictlyBinomial)`
- `#define PB_DS_B_HEAP_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 5.41.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.42 bit File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_BIT`

## 5.42.1 Detailed Description

This is a Standard C++ Library header.

## 5.43 bitmap\_allocator.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_mini\\_vector<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Bitmap\\_counter<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_detail::\\_\\_Ffit\\_finder<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::bitmap\\_allocator<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::free\\_list](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::\\_\\_detail](#)

## Macros

- `#define \_BALLOC\_ALIGN\_BYTES`

## Enumerations

- `enum { bits\_per\_byte, bits\_per\_block }`

## Functions

- `void \_\_gnu\_cxx::\_\_detail::\_\_bit\_allocate (std::size_t *__pmap, std::size_t __pos) throw ()`
- `void \_\_gnu\_cxx::\_\_detail::\_\_bit\_free (std::size_t *__pmap, std::size_t __pos) throw ()`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >  
_ForwardIterator \_\_gnu\_cxx::\_\_detail::\_\_lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const  
_Tp &__val, _Compare __comp)`
- `template<typename _AddrPair >  
std::size_t \_\_gnu\_cxx::\_\_detail::\_\_num\_bitmaps (_AddrPair __ap)`
- `template<typename _AddrPair >  
std::size_t \_\_gnu\_cxx::\_\_detail::\_\_num\_blocks (_AddrPair __ap)`
- `std::size_t \_\_gnu\_cxx::\_\_Bit\_scan\_forward (std::size_t __num)`
- `template<typename _Tp1, typename _Tp2 >  
bool \_\_gnu\_cxx::operator!= (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`
- `template<typename _Tp1, typename _Tp2 >  
bool \_\_gnu\_cxx::operator== (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()`

### 5.43.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.43.2 Macro Definition Documentation

#### 5.43.2.1 [\\_BALLOC\\_ALIGN\\_BYTES](#)

```
#define _BALLOC_ALIGN_BYTES
```

The constant in the expression below is the alignment required in bytes.

Definition at line 43 of file `bitmap_allocator.h`.

## 5.44 **bitset** File Reference

### Classes

- struct [std::\\_Base\\_bitset< \\_Nw >](#)
- struct [std::\\_Base\\_bitset< 0 >](#)
- struct [std::\\_Base\\_bitset< 1 >](#)
- class [std::bitset< \\_Nb >](#)
- struct [std::hash<::bitset< \\_Nb > >](#)
- class [std::bitset< \\_Nb >::reference](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBCXX\_BITSET`
- `#define \_GLIBCXX\_BITSET\_BITS\_PER\_ULL`
- `#define \_GLIBCXX\_BITSET\_BITS\_PER\_WORD`
- `#define \_GLIBCXX\_BITSET\_WORDS\(\_\_n\)`

### Functions

- `template<size_t _Nb>`  
`bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<size_t _Nb>`  
`bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_istream< _CharT, _Traits > & std::operator>> (std::basic\_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)`
- `template<class _CharT, class _Traits, size_t _Nb>`  
`std::basic\_ostream< _CharT, _Traits > & std::operator<< (std::basic\_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)`

#### 5.44.1 Detailed Description

This is a Standard C++ Library header.

## 5.45 bitset File Reference

### Classes

- class [std::\\_\\_debug::bitset<\\_Nb>](#)
- struct [std::hash<\\_\\_debug::bitset<\\_Nb>>](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

### Functions

- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator& (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_ostream<_CharT, _Traits> & std::__debug::operator<< (std::basic\_ostream<_CharT, _Traits> &__os, const bitset<_Nb> &__x)`
- `template<typename _CharT, typename _Traits, size_t _Nb>  
std::basic\_istream<_CharT, _Traits> & std::__debug::operator>> (std::basic\_istream<_CharT, _Traits> &__is, bitset<_Nb> &__x)`
- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator^ (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`
- `template<size_t _Nb>  
bitset<_Nb> std::__debug::operator| (const bitset<_Nb> &__x, const bitset<_Nb> &__y) noexcept`

### 5.45.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.46 bool\_set File Reference

### Classes

- class [std::tr2::bool\\_set](#)

### Namespaces

- [std](#)
- [std::tr2](#)

### Macros

- `#define _GLIBCXX_TR2_BOOL_SET`

## Functions

- bool **std::tr2::certainly** (bool\_set \_\_b)
- bool **std::tr2::contains** (bool\_set \_\_s, bool\_set \_\_t)
- bool **std::tr2::equals** (bool\_set \_\_s, bool\_set \_\_t)
- bool **std::tr2::is\_emptyset** (bool\_set \_\_b)
- bool **std::tr2::is\_indeterminate** (bool\_set \_\_b)
- bool **std::tr2::is\_singleton** (bool\_set \_\_b)
- bool\_set **std::tr2::operator!=** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator!=** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::operator!=** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator&** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator&** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::operator==** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator==** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::operator^** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator^** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::operator|** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::operator|** (bool\_set \_\_s, bool \_\_t)
- bool **std::tr2::possibly** (bool\_set \_\_b)
- bool\_set **std::tr2::set\_complement** (bool\_set \_\_b)
- bool\_set **std::tr2::set\_intersection** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::set\_intersection** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::set\_intersection** (bool\_set \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::set\_union** (bool \_\_s, bool\_set \_\_t)
- bool\_set **std::tr2::set\_union** (bool\_set \_\_s, bool \_\_t)
- bool\_set **std::tr2::set\_union** (bool\_set \_\_s, bool\_set \_\_t)

## 5.46.1 Detailed Description

This is a TR2 C++ Library header.

## 5.47 bool\_set.tcc File Reference

## Namespaces

- [std](#)
- [std::tr2](#)

## Macros

- `#define GLIBCXX_TR2_BOOL_SET_TCC`

## 5.47.1 Detailed Description

This is a TR2 C++ Library header.

## 5.48 boost\_concept\_check.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define GLIBCXX_CLASS_REQUIRES(_type_var, _ns, _concept)`
- `#define GLIBCXX_CLASS_REQUIRES2(_type_var1, _type_var2, _ns, _concept)`
- `#define GLIBCXX_CLASS_REQUIRES3(_type_var1, _type_var2, _type_var3, _ns, _concept)`
- `#define GLIBCXX_CLASS_REQUIRES4(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)`
- `#define GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT(_OP, _NAME)`
- `#define GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT(_OP, _NAME)`
- `#define _IsUnused`

### Functions

- `template<class _Tp >`  
`void __gnu_cxx::__aux_require_boolean_expr (const _Tp &__t)`
- `void __gnu_cxx::__error_type_must_be_a_signed_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_integer_type ()`
- `void __gnu_cxx::__error_type_must_be_an_unsigned_integer_type ()`
- `template<class _Concept >`  
`constexpr void __gnu_cxx::__function_requires ()`

#### 5.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.49 branch\_policy.hpp File Reference

### Classes

- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc >`
- `struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >`

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.49.1 Detailed Description

Contains a base class for branch policies.



## 5.50 `c++0x_warning.h` File Reference

### 5.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.51 `c++allocator.h` File Reference

### Namespaces

- [std](#)

### Typedefs

- `template<typename _Tp >`  
`using std::\_\_allocator\_base = \_\_gnu\_cxx::new\_allocator<_Tp >`

### 5.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.52 `c++config.h` File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define` `__GLIBCXX__`
- `#define` `__glibcxx_assert`(\_Condition)
- `#define` `__N`(msgid)
- `#define` `_GLIBCXX11_USE_C99_MATH`
- `#define` `_GLIBCXX11_USE_C99_STDIO`
- `#define` `_GLIBCXX11_USE_C99_STDLIB`
- `#define` `_GLIBCXX11_USE_C99_WCHAR`
- `#define` `_GLIBCXX17_DEPRECATED`
- `#define` `_GLIBCXX20_DEPRECATED`(MSG)
- `#define` `_GLIBCXX98_USE_C99_COMPLEX`
- `#define` `_GLIBCXX98_USE_C99_MATH`
- `#define` `_GLIBCXX98_USE_C99_STDIO`

- `#define _GLIBCXX98_USE_C99_STDLIB`
- `#define _GLIBCXX98_USE_C99_WCHAR`
- `#define _GLIBCXX_ABI_TAG_CXX11`
- `#define _GLIBCXX_ATOMIC_BUILTINS`
- `#define _GLIBCXX_BEGIN_EXTERN_C`
- `#define _GLIBCXX_BEGIN_NAMESPACE_ALGO`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_BEGIN_NAMESPACE_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL`
- `#define _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_BEGIN_NAMESPACE_VERSION`
- `#define _GLIBCXX_DARWIN_USE_64_BIT_INODE`
- `#define _GLIBCXX_DEFAULT_ABI_TAG`
- `#define _GLIBCXX_DEPRECATED`
- `#define _GLIBCXX_DEPRECATED_SUGGEST(ALT)`
- `#define _GLIBCXX_END_EXTERN_C`
- `#define _GLIBCXX_END_NAMESPACE_ALGO`
- `#define _GLIBCXX_END_NAMESPACE_CONTAINER`
- `#define _GLIBCXX_END_NAMESPACE_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_LDBL`
- `#define _GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_END_NAMESPACE_VERSION`
- `#define _GLIBCXX_EXTERN_TEMPLATE`
- `#define _GLIBCXX_FAST_MATH`
- `#define _GLIBCXX_FULLY_DYNAMIC_STRING`
- `#define _GLIBCXX_HAVE__CXA_THREAD_ATEXIT_IMPL`
- `#define _GLIBCXX_HAVE_ACOSF`
- `#define _GLIBCXX_HAVE_ACOSL`
- `#define _GLIBCXX_HAVE_ALIGNED_ALLOC`
- `#define _GLIBCXX_HAVE_ARPA_INET_H`
- `#define _GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE`
- `#define _GLIBCXX_HAVE_ASINF`
- `#define _GLIBCXX_HAVE_ASINL`
- `#define _GLIBCXX_HAVE_AT_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_ATAN2F`
- `#define _GLIBCXX_HAVE_ATAN2L`
- `#define _GLIBCXX_HAVE_ATANF`
- `#define _GLIBCXX_HAVE_ATANL`
- `#define _GLIBCXX_HAVE_ATOMIC_LOCK_POLICY`
- `#define _GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY`
- `#define _GLIBCXX_HAVE_CEILF`
- `#define _GLIBCXX_HAVE_CEILL`
- `#define _GLIBCXX_HAVE_COMPLEX_H`
- `#define _GLIBCXX_HAVE_COSF`
- `#define _GLIBCXX_HAVE_COSHF`
- `#define _GLIBCXX_HAVE_COSHL`
- `#define _GLIBCXX_HAVE_COSL`
- `#define _GLIBCXX_HAVE_DIRENT_H`
- `#define _GLIBCXX_HAVE_DLFCN_H`
- `#define _GLIBCXX_HAVE_ENDIAN_H`
- `#define _GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46`

- `#define _GLIBCXX_HAVE_EXECINFO_H`
- `#define _GLIBCXX_HAVE_EXPF`
- `#define _GLIBCXX_HAVE_EXPL`
- `#define _GLIBCXX_HAVE_FABSF`
- `#define _GLIBCXX_HAVE_FABSL`
- `#define _GLIBCXX_HAVE_FCNTL_H`
- `#define _GLIBCXX_HAVE_FENV_H`
- `#define _GLIBCXX_HAVE_FINITE`
- `#define _GLIBCXX_HAVE_FINITEF`
- `#define _GLIBCXX_HAVE_FINITEL`
- `#define _GLIBCXX_HAVE_FLOAT_H`
- `#define _GLIBCXX_HAVE_FLOORF`
- `#define _GLIBCXX_HAVE_FLOORL`
- `#define _GLIBCXX_HAVE_FMODF`
- `#define _GLIBCXX_HAVE_FMODL`
- `#define _GLIBCXX_HAVE_FREXPF`
- `#define _GLIBCXX_HAVE_FREXPL`
- `#define _GLIBCXX_HAVE_GETIPINFO`
- `#define _GLIBCXX_HAVE_GETS`
- `#define _GLIBCXX_HAVE_HYPOT`
- `#define _GLIBCXX_HAVE_HYPOTF`
- `#define _GLIBCXX_HAVE_HYPOTL`
- `#define _GLIBCXX_HAVE_ICONv`
- `#define _GLIBCXX_HAVE_INT64_T`
- `#define _GLIBCXX_HAVE_INT64_T_LONG`
- `#define _GLIBCXX_HAVE_INTTYPES_H`
- `#define _GLIBCXX_HAVE_ISINF`
- `#define _GLIBCXX_HAVE_ISINFF`
- `#define _GLIBCXX_HAVE_ISINFL`
- `#define _GLIBCXX_HAVE_ISNAN`
- `#define _GLIBCXX_HAVE_ISNANF`
- `#define _GLIBCXX_HAVE_ISNANL`
- `#define _GLIBCXX_HAVE_ISWBLANK`
- `#define _GLIBCXX_HAVE_LC_MESSAGES`
- `#define _GLIBCXX_HAVE_LDEXPF`
- `#define _GLIBCXX_HAVE_LDEXPL`
- `#define _GLIBCXX_HAVE_LIMIT_AS`
- `#define _GLIBCXX_HAVE_LIMIT_DATA`
- `#define _GLIBCXX_HAVE_LIMIT_FSIZE`
- `#define _GLIBCXX_HAVE_LIMIT_RSS`
- `#define _GLIBCXX_HAVE_LIMIT_VMEM`
- `#define _GLIBCXX_HAVE_LINK`
- `#define _GLIBCXX_HAVE_LINUX_FUTEX`
- `#define _GLIBCXX_HAVE_LINUX_RANDOM_H`
- `#define _GLIBCXX_HAVE_LINUX_TYPES_H`
- `#define _GLIBCXX_HAVE_LOCALE_H`
- `#define _GLIBCXX_HAVE_LOG10F`
- `#define _GLIBCXX_HAVE_LOG10L`
- `#define _GLIBCXX_HAVE_LOGF`
- `#define _GLIBCXX_HAVE_LOGL`
- `#define _GLIBCXX_HAVE_MBSTATE_T`

- `#define _GLIBCXX_HAVE_MEMALIGN`
- `#define _GLIBCXX_HAVE_MEMORY_H`
- `#define _GLIBCXX_HAVE_MODFF`
- `#define _GLIBCXX_HAVE_MODFL`
- `#define _GLIBCXX_HAVE_NETDB_H`
- `#define _GLIBCXX_HAVE_NETINET_IN_H`
- `#define _GLIBCXX_HAVE_NETINET_TCP_H`
- `#define _GLIBCXX_HAVE_POLL`
- `#define _GLIBCXX_HAVE_POLL_H`
- `#define _GLIBCXX_HAVE_POSIX_MEMALIGN`
- `#define _GLIBCXX_HAVE_POWF`
- `#define _GLIBCXX_HAVE_POWL`
- `#define _GLIBCXX_HAVE_QUICK_EXIT`
- `#define _GLIBCXX_HAVE_READLINK`
- `#define _GLIBCXX_HAVE_S_ISREG`
- `#define _GLIBCXX_HAVE_SETENV`
- `#define _GLIBCXX_HAVE_SINCOS`
- `#define _GLIBCXX_HAVE_SINCOSF`
- `#define _GLIBCXX_HAVE_SINCOSL`
- `#define _GLIBCXX_HAVE_SINF`
- `#define _GLIBCXX_HAVE_SINHF`
- `#define _GLIBCXX_HAVE_SINHL`
- `#define _GLIBCXX_HAVE_SINL`
- `#define _GLIBCXX_HAVE_SOCKETMARK`
- `#define _GLIBCXX_HAVE_SQRTF`
- `#define _GLIBCXX_HAVE_SQRTL`
- `#define _GLIBCXX_HAVE_STDALIGN_H`
- `#define _GLIBCXX_HAVE_STDBOOL_H`
- `#define _GLIBCXX_HAVE_STDINT_H`
- `#define _GLIBCXX_HAVE_STDLIB_H`
- `#define _GLIBCXX_HAVE_STRERROR_L`
- `#define _GLIBCXX_HAVE_STRERROR_R`
- `#define _GLIBCXX_HAVE_STRING_H`
- `#define _GLIBCXX_HAVE_STRINGS_H`
- `#define _GLIBCXX_HAVE_STRTOF`
- `#define _GLIBCXX_HAVE_STRTOLD`
- `#define _GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE`
- `#define _GLIBCXX_HAVE_STRXFRM_L`
- `#define _GLIBCXX_HAVE_SYMLINK`
- `#define _GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT`
- `#define _GLIBCXX_HAVE_SYS_IOCTL_H`
- `#define _GLIBCXX_HAVE_SYS_IPC_H`
- `#define _GLIBCXX_HAVE_SYS_PARAM_H`
- `#define _GLIBCXX_HAVE_SYS_RESOURCE_H`
- `#define _GLIBCXX_HAVE_SYS_SEM_H`
- `#define _GLIBCXX_HAVE_SYS_SOCKET_H`
- `#define _GLIBCXX_HAVE_SYS_STAT_H`
- `#define _GLIBCXX_HAVE_SYS_STATVFS_H`
- `#define _GLIBCXX_HAVE_SYS_SYSINFO_H`
- `#define _GLIBCXX_HAVE_SYS_TIME_H`
- `#define _GLIBCXX_HAVE_SYS_TYPES_H`

- `#define _GLIBCXX_HAVE_SYS_UIO_H`
- `#define _GLIBCXX_HAVE_TANF`
- `#define _GLIBCXX_HAVE_TANHF`
- `#define _GLIBCXX_HAVE_TANHL`
- `#define _GLIBCXX_HAVE_TANL`
- `#define _GLIBCXX_HAVE_TGMATH_H`
- `#define _GLIBCXX_HAVE_TIMESPEC_GET`
- `#define _GLIBCXX_HAVE_TLS`
- `#define _GLIBCXX_HAVE_TRUNCATE`
- `#define _GLIBCXX_HAVE_UCHAR_H`
- `#define _GLIBCXX_HAVE_UNISTD_H`
- `#define _GLIBCXX_HAVE_UTIME_H`
- `#define _GLIBCXX_HAVE_VFWSCANF`
- `#define _GLIBCXX_HAVE_VSWSCANF`
- `#define _GLIBCXX_HAVE_VWSCANF`
- `#define _GLIBCXX_HAVE_WCHAR_H`
- `#define _GLIBCXX_HAVE_WCSTOF`
- `#define _GLIBCXX_HAVE_WCTYPE_H`
- `#define _GLIBCXX_HAVE_WRITEV`
- `#define _GLIBCXX_HOSTED`
- `#define _GLIBCXX_ICONV_CONST`
- `#define _GLIBCXX_INLINE_VERSION`
- `#define _GLIBCXX_MANGLE_SIZE_T`
- `#define _GLIBCXX_NAMESPACE_CXX11`
- `#define _GLIBCXX_NAMESPACE_LDBL`
- `#define _GLIBCXX_NAMESPACE_LDBL_OR_CXX11`
- `#define _GLIBCXX_NODISCARD`
- `#define _GLIBCXX_NOEXCEPT_PARM`
- `#define _GLIBCXX_NOEXCEPT_QUAL`
- `#define _GLIBCXX_PACKAGE__GLIBCXX_VERSION`
- `#define _GLIBCXX_PACKAGE_BUGREPORT`
- `#define _GLIBCXX_PACKAGE_NAME`
- `#define _GLIBCXX_PACKAGE_STRING`
- `#define _GLIBCXX_PACKAGE_TARNAME`
- `#define _GLIBCXX_PACKAGE_URL`
- `#define _GLIBCXX_PSEUDO_VISIBILITY(V)`
- `#define _GLIBCXX_RELEASE`
- `#define _GLIBCXX_RES_LIMITS`
- `#define _GLIBCXX_STD_A`
- `#define _GLIBCXX_STD_C`
- `#define _GLIBCXX_STDIO_EOF`
- `#define _GLIBCXX_STDIO_SEEK_CUR`
- `#define _GLIBCXX_STDIO_SEEK_END`
- `#define _GLIBCXX_SYMVER`
- `#define _GLIBCXX_SYMVER_GNU`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER(A)`
- `#define _GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE(A)`
- `#define _GLIBCXX_THROW_OR_ABORT(_EXC)`
- `#define _GLIBCXX_TXN_SAFE`
- `#define _GLIBCXX_TXN_SAFE_DYN`
- `#define _GLIBCXX_USE_ALLOCATOR_NEW`

- `#define _GLIBCXX_USE_C11_UCHAR_CXX11`
- `#define _GLIBCXX_USE_C99`
- `#define _GLIBCXX_USE_C99_COMPLEX`
- `#define _GLIBCXX_USE_C99_COMPLEX_TR1`
- `#define _GLIBCXX_USE_C99_CTYPE_TR1`
- `#define _GLIBCXX_USE_C99_FENV_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_TR1`
- `#define _GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1`
- `#define _GLIBCXX_USE_C99_MATH`
- `#define _GLIBCXX_USE_C99_MATH_TR1`
- `#define _GLIBCXX_USE_C99_STDINT_TR1`
- `#define _GLIBCXX_USE_C99_STDIO`
- `#define _GLIBCXX_USE_C99_STDLIB`
- `#define _GLIBCXX_USE_C99_WCHAR`
- `#define _GLIBCXX_USE_CLOCK_MONOTONIC`
- `#define _GLIBCXX_USE_CLOCK_REALTIME`
- `#define _GLIBCXX_USE_CXX11_ABI`
- `#define _GLIBCXX_USE_DECIMAL_FLOAT`
- `#define _GLIBCXX_USE_DEPRECATED`
- `#define _GLIBCXX_USE_DEV_RANDOM`
- `#define _GLIBCXX_USE_DUAL_ABI`
- `#define _GLIBCXX_USE_FCHMOD`
- `#define _GLIBCXX_USE_FCHMODAT`
- `#define _GLIBCXX_USE_GET_NPROCS`
- `#define _GLIBCXX_USE_GETTIMEOFDAY`
- `#define _GLIBCXX_USE_INT128`
- `#define _GLIBCXX_USE_LFS`
- `#define _GLIBCXX_USE_LONG_LONG`
- `#define _GLIBCXX_USE_LSTAT`
- `#define _GLIBCXX_USE_NANOSLEEP`
- `#define _GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT`
- `#define _GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK`
- `#define _GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK`
- `#define _GLIBCXX_USE_RANDOM_TR1`
- `#define _GLIBCXX_USE_REALPATH`
- `#define _GLIBCXX_USE_SC_NPROCESSORS_ONLN`
- `#define _GLIBCXX_USE_SCHED_YIELD`
- `#define _GLIBCXX_USE_SENDFILE`
- `#define _GLIBCXX_USE_ST_MTIM`
- `#define _GLIBCXX_USE_TMPNAM`
- `#define _GLIBCXX_USE_UTIME`
- `#define _GLIBCXX_USE_UTIMENSAT`
- `#define _GLIBCXX_USE_WCHAR_T`
- `#define _GLIBCXX_USE_WEAK_REF`
- `#define _GLIBCXX_VERBOSE`
- `#define _GLIBCXX_VISIBILITY(V)`
- `#define _GLIBCXX_WEAK_DEFINITION`
- `#define _GLIBCXX_X86_RDRAND`
- `#define _GLIBCXX_X86_RDSEED`
- `#define _GTHREAD_USE_MUTEX_TIMEDLOCK`
- `#define LT_OBJDIR`
- `#define STDC_HEADERS`

## Typedefs

- typedef `__PTRDIFF_TYPE__` **std::ptrdiff\_t**
- typedef `__SIZE_TYPE__` **std::size\_t**

## Variables

- decltype(nullptr) typedef **std::nullptr\_t**

## 5.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<version>`.

5.53 `c++io.h` File Reference

## Namespaces

- [std](#)

## Typedefs

- typedef FILE **std::\_\_c\_file**
- typedef `__gthread_mutex_t` **std::\_\_c\_lock**

## 5.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

5.54 `c++locale.h` File Reference

## Namespaces

- [std](#)

## Macros

- #define **\_GLIBCXX\_C\_LOCALE\_GNU**
- #define **\_GLIBCXX\_NUM\_CATEGORIES**

### Typedefs

- typedef `__locale_t` **std::\_\_c\_locale**

### Functions

- int **std::\_\_convert\_from\_v** (const `__c_locale` &`__cloc`, char \*`__out`, const int `__size`, const char \*`__fmt`,...)

#### 5.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.55 `c++locale_internal.h` File Reference

### Namespaces

- [std](#)

### Functions

- Catalogs & **std::get\_catalogs** ()

#### 5.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.56 `cassert` File Reference

#### 5.56.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.57 `cast.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Caster<\\_ToType>](#)



## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_const\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_dynamic\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_reinterpret\_pointer\_cast (_FromType *__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (const _FromType &__arg)`
- `template<typename _ToType, typename _FromType >  
_ToType \_\_gnu\_cxx::\_\_static\_pointer\_cast (_FromType *__arg)`

## 5.57.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/pointer.h>`.

## 5.58 cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp File Reference

## 5.58.1 Detailed Description

Contains a resize trigger implementation.

## 5.59 cc\_ht\_map.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map](#)< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CC_HASH_NAME`
- `#define PB_DS_CC_HASH_TRAITS_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_HASH_FN_C_DEC`

### 5.59.1 Detailed Description

Contains an implementation class for `cc_ht_map_`.

## 5.60 `ccomplex` File Reference

### Macros

- `#define _GLIBCXX_CCOMPLEX`

### 5.60.1 Detailed Description

This is a Standard C++ Library header.

## 5.61 `ccomplex` File Reference

### Macros

- `#define _GLIBCXX_TR1_CCOMPLEX`

### 5.61.1 Detailed Description

This is a TR1 C++ Library header.

## 5.62 `cctype` File Reference

### Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_CCTYPE`

## 5.62.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

5.63 `cctype` File Reference

## Macros

- `#define _GLIBCXX_TR1_CCTYPE`

## 5.63.1 Detailed Description

This is a TR1 C++ Library header.

5.64 `cerrno` File Reference

## Macros

- `#define _GLIBCXX_CERRNO`
- `#define errno`

## 5.64.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

5.65 `cfenv` File Reference

## Macros

- `#define _GLIBCXX_CFENV`

#### 5.65.1 Detailed Description

This is a Standard C++ Library header.

### 5.66 `cfenv` File Reference

#### Macros

- `#define _GLIBCXX_TR1_CFENV`

#### 5.66.1 Detailed Description

This is a TR1 C++ Library header.

### 5.67 `cfloat` File Reference

#### Macros

- `#define _GLIBCXX_CFLOAT`
- `#define DECIMAL_DIG`
- `#define FLT_EVAL_METHOD`

#### 5.67.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 5.68 `cfloat` File Reference

#### Macros

- `#define _GLIBCXX_TR1_CFLOAT`

#### 5.68.1 Detailed Description

This is a TR1 C++ Library header.

## 5.69 char\_traits.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::\\_Char\\_types<\\_CharT>](#)
- struct [std::char\\_traits<\\_CharT>](#)
- struct [\\_\\_gnu\\_cxx::char\\_traits<\\_CharT>](#)
- struct [std::char\\_traits<char>](#)
- struct [std::char\\_traits<wchar\\_t>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define \_GLIBCXX\_ALWAYS\_INLINE`

#### 5.69.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

## 5.70 charconv File Reference

### Classes

- struct [std::from\\_chars\\_result](#)
- struct [std::to\\_chars\\_result](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Macros

- `#define \_GLIBCXX\_CHARCONV`
- `#define \_GLIBCXX\_TO\_CHARS\(T\)`

## Typedefs

- `template<typename _Tp >`  
`using std::__detail::__integer_from_chars_result_type = enable_if_t<__or_<__is_signed_integer<_Tp>, __is_unsigned_integer<_Tp>, is_same<char, remove_cv_t<_Tp>>>::value, from_chars_result >`
- `template<typename _Tp >`  
`using std::__detail::__integer_to_chars_result_type = enable_if_t<__or_<__is_signed_integer<_Tp>, ← __is_unsigned_integer<_Tp>, is_same<char, remove_cv_t<_Tp>>>::value, to_chars_result >`
- `template<typename _Tp >`  
`using std::__detail::__unsigned_least_t = typename __to_chars_unsigned_type<_Tp>::type`

## Enumerations

- `enum std::chars_format { scientific, fixed, hex, general }`

## Functions

- `template<typename _Tp >`  
`bool std::__detail::__from_chars_alnum (const char * __first, const char * __last, _Tp & __val, int __base)`
- `constexpr unsigned char std::__detail::__from_chars_alpha_to_num (char __c)`
- `template<typename _Tp >`  
`bool std::__detail::__from_chars_binary (const char * __first, const char * __last, _Tp & __val)`
- `template<typename _Tp >`  
`bool std::__detail::__from_chars_digit (const char * __first, const char * __last, _Tp & __val, int __base)`
- `template<typename _Tp >`  
`bool std::__detail::__raise_and_add (_Tp & __val, int __base, unsigned char __c)`
- `template<typename _Tp >`  
`to_chars_result std::__detail::__to_chars (char * __first, char * __last, _Tp __val, int __base) noexcept`
- `template<typename _Tp >`  
`__integer_to_chars_result_type<_Tp> std::__detail::__to_chars_10 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp >`  
`__integer_to_chars_result_type<_Tp> std::__detail::__to_chars_16 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp >`  
`__integer_to_chars_result_type<_Tp> std::__detail::__to_chars_2 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp >`  
`__integer_to_chars_result_type<_Tp> std::__detail::__to_chars_8 (char * __first, char * __last, _Tp __val) noexcept`
- `template<typename _Tp >`  
`__detail::__integer_to_chars_result_type<_Tp> std::__to_chars_i (char * __first, char * __last, _Tp __value, int __base=10)`
- `template<typename _Tp >`  
`constexpr unsigned std::__detail::__to_chars_len (_Tp __value, int __base) noexcept`
- `template<typename _Tp >`  
`constexpr unsigned std::__detail::__to_chars_len_2 (_Tp __value) noexcept`
- `template<typename _Tp >`  
`__detail::__integer_from_chars_result_type<_Tp> std::from_chars (const char * __first, const char * __last, _Tp & __value, int __base=10)`
- `constexpr chars_format std::operator& (chars_format __lhs, chars_format __rhs) noexcept`

- constexpr chars\_format & **std::operator&=** (chars\_format \_\_lhs, chars\_format \_\_rhs) noexcept
- constexpr chars\_format **std::operator^** (chars\_format \_\_lhs, chars\_format \_\_rhs) noexcept
- constexpr chars\_format & **std::operator^=** (chars\_format \_\_lhs, chars\_format \_\_rhs) noexcept
- constexpr chars\_format **std::operator|** (chars\_format \_\_lhs, chars\_format \_\_rhs) noexcept
- constexpr chars\_format & **std::operator|=** (chars\_format \_\_lhs, chars\_format \_\_rhs) noexcept
- constexpr chars\_format **std::operator~** (chars\_format \_\_fmt) noexcept
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, char \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, signed char \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, unsigned char \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, signed short \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, unsigned short \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, signed int \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, unsigned int \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, signed long \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, unsigned long \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, signed long long \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \* \_\_first, char \* \_\_last, unsigned long long \_\_value, int \_\_base=10)
- to\_chars\_result **std::to\_chars** (char \*, char \*, bool, int=10)=delete

#### 5.70.1 Detailed Description

This is a Standard C++ Library header.

## 5.71 charconv.h File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Functions

- template<typename \_Tp >  
void **std::\_\_detail::\_\_to\_chars\_10\_impl** (char \* \_\_first, unsigned \_\_len, \_Tp \_\_val) noexcept
- template<typename \_Tp >  
constexpr unsigned **std::\_\_detail::\_\_to\_chars\_len** (\_Tp \_\_value, int \_\_base) noexcept

#### 5.71.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<charconv>`.

## 5.72 checkers.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _Iter , typename _Compare >`  
`bool \_\_gnu\_parallel::\_\_is\_sorted (_Iter __begin, _Iter __end, _Compare __comp)`

### 5.72.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

## 5.73 chrono File Reference

### Classes

- `struct std::common\_type< chrono::duration< \_Rep, \_Period > >`
- `struct std::common\_type< chrono::duration< \_Rep, \_Period >, chrono::duration< \_Rep, \_Period > >`
- `struct std::common\_type< chrono::duration< \_Rep1, \_Period1 >, chrono::duration< \_Rep2, \_Period2 > >`
- `struct std::common\_type< chrono::time\_point< \_Clock, \_Duration > >`
- `struct std::common\_type< chrono::time\_point< \_Clock, \_Duration >, chrono::time\_point< \_Clock, \_Duration > >`
- `struct std::common\_type< chrono::time\_point< \_Clock, \_Duration1 >, chrono::time\_point< \_Clock, \_Duration2 > >`
- `struct std::chrono::duration< \_Rep, \_Period >`
- `struct std::chrono::duration< \_Rep, \_Period >`
- `struct std::chrono::duration\_values< \_Rep >`
- `struct std::chrono::\_V2::steady\_clock`
- `struct std::chrono::\_V2::system\_clock`
- `struct std::chrono::time\_point< \_Clock, \_Dur >`
- `struct std::chrono::time\_point< \_Clock, \_Dur >`
- `struct std::chrono::treat\_as\_floating\_point< \_Rep >`

### Namespaces

- `std`
- `std::chrono`
- `std::literals::chrono\_literals`

### Macros

- `#define \_\_cpp\_lib\_chrono\_udls`
- `#define \_GLIBCXX\_CHRONO`
- `#define \_GLIBCXX\_CHRONO\_INT64\_T`



### Typedefs

- using [std::chrono::\\_V2::high\\_resolution\\_clock](#) = system\_clock
- using [std::chrono::hours](#) = duration< int64\_t, ratio< 3600 > >
- using [std::chrono::microseconds](#) = duration< int64\_t, micro >
- using [std::chrono::milliseconds](#) = duration< int64\_t, milli >
- using [std::chrono::minutes](#) = duration< int64\_t, ratio< 60 > >
- using [std::chrono::nanoseconds](#) = duration< int64\_t, nano >
- using [std::chrono::seconds](#) = duration< int64\_t >

### Functions

- template<typename \_ToDur , typename \_Rep , typename \_Period >  
constexpr \_\_enable\_if\_is\_duration< \_ToDur > [std::chrono::duration\\_cast](#) (const duration< \_Rep, \_Period > &←  
\_\_d)
- constexpr chrono::duration< long double, ratio< 3600, 1 > > [std::literals::chrono\\_literals::operator""h](#) (long double \_\_hours)
- template<char... \_Digits>  
constexpr chrono::hours [std::literals::chrono\\_literals::operator""h](#) ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > [std::literals::chrono\\_literals::operator""min](#) (long double \_\_mins)
- template<char... \_Digits>  
constexpr chrono::minutes [std::literals::chrono\\_literals::operator""min](#) ()
- constexpr chrono::duration< long double, milli > [std::literals::chrono\\_literals::operator""ms](#) (long double \_\_←  
msecs)
- template<char... \_Digits>  
constexpr chrono::milliseconds [std::literals::chrono\\_literals::operator""ms](#) ()
- constexpr chrono::duration< long double, nano > [std::literals::chrono\\_literals::operator""ns](#) (long double \_\_nsecs)
- template<char... \_Digits>  
constexpr chrono::nanoseconds [std::literals::chrono\\_literals::operator""ns](#) ()
- constexpr chrono::duration< long double > [std::literals::chrono\\_literals::operator""s](#) (long double \_\_secs)
- template<char... \_Digits>  
constexpr chrono::seconds [std::literals::chrono\\_literals::operator""s](#) ()
- constexpr chrono::duration< long double, micro > [std::literals::chrono\\_literals::operator""us](#) (long double \_\_←  
usecs)
- template<char... \_Digits>  
constexpr chrono::microseconds [std::literals::chrono\\_literals::operator""us](#) ()
- template<typename \_ToDur , typename \_Clock , typename \_Dur >  
constexpr enable\_if< \_\_is\_duration< \_ToDur >::value, time\_point< \_Clock, \_ToDur > >::type [std::chrono::time\\_point\\_cast](#)  
(const time\_point< \_Clock, \_Dur > &\_\_t)

#### 5.73.1 Detailed Description

This is a Standard C++ Library header.

## 5.74 chrono File Reference

### Namespaces

- [std](#)
- [std::chrono](#)

#### Macros

- `#define _GLIBCXX_EXPERIMENTAL_CHRONO`

#### Variables

- `template<typename _Rep >`  
`constexpr bool std::chrono::experimental::fundamentals_v1::treat_as_floating_point_v`

#### 5.74.1 Detailed Description

This is a TS C++ Library header.

### 5.75 cinttypes File Reference

#### Macros

- `#define _GLIBCXX_CINTTYPES`

#### 5.75.1 Detailed Description

This is a Standard C++ Library header.

### 5.76 cinttypes File Reference

#### Macros

- `#define _GLIBCXX_TR1_CINTTYPES`

#### 5.76.1 Detailed Description

This is a TR1 C++ Library header.

### 5.77 ciso646 File Reference

#### 5.77.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

## 5.78 climits File Reference

### Macros

- `#define _GLIBCXX_CLIMITS`
- `#define LLONG_MAX`
- `#define LLONG_MIN`
- `#define ULLONG_MAX`

#### 5.78.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.79 climits File Reference

### Macros

- `#define _GLIBCXX_TR1_CLIMITS`

#### 5.79.1 Detailed Description

This is a TR1 C++ Library header.

## 5.80 clocale File Reference

### Namespaces

- `std`

### Macros

- `#define _GLIBCXX_CLOCALE`

#### 5.80.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.81 cmath File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CMATH`
- `#define _GLIBCXX_INCLUDE_NEXT_C_HEADERS`

### Functions

- `constexpr float std::acos (float __x)`
- `constexpr long double std::acos (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::acos (_Tp __x)`
- `constexpr float std::asin (float __x)`
- `constexpr long double std::asin (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::asin (_Tp __x)`
- `constexpr float std::atan (float __x)`
- `constexpr long double std::atan (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::atan (_Tp __x)`
- `constexpr float std::atan2 (float __y, float __x)`
- `constexpr long double std::atan2 (long double __y, long double __x)`
- `template<typename _Tp, typename _Up >`  
`constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type std::atan2 (_Tp __y, _Up __x)`
- `constexpr float std::ceil (float __x)`
- `constexpr long double std::ceil (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::ceil (_Tp __x)`
- `constexpr float std::cos (float __x)`
- `constexpr long double std::cos (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cos (_Tp __x)`
- `constexpr float std::cosh (float __x)`
- `constexpr long double std::cosh (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::cosh (_Tp __x)`
- `constexpr float std::exp (float __x)`
- `constexpr long double std::exp (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::exp (_Tp __x)`
- `constexpr float std::fabs (float __x)`
- `constexpr long double std::fabs (long double __x)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type std::fabs (_Tp __x)`

- constexpr float **std::floor** (float \_\_x)
- constexpr long double **std::floor** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::floor** (\_Tp \_\_x)
- constexpr float **std::fmod** (float \_\_x, float \_\_y)
- constexpr long double **std::fmod** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::fmod** (\_Tp \_\_x, \_Up \_\_y)
- float **std::frexp** (float \_\_x, int \*\_\_exp)
- long double **std::frexp** (long double \_\_x, int \*\_\_exp)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::frexp** (\_Tp \_\_x, int \*\_\_exp)
- constexpr float **std::ldexp** (float \_\_x, int \_\_exp)
- constexpr long double **std::ldexp** (long double \_\_x, int \_\_exp)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::ldexp** (\_Tp \_\_x, int \_\_exp)
- constexpr float **std::log** (float \_\_x)
- constexpr long double **std::log** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log** (\_Tp \_\_x)
- constexpr float **std::log10** (float \_\_x)
- constexpr long double **std::log10** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::log10** (\_Tp \_\_x)
- float **std::modf** (float \_\_x, float \*\_\_iptr)
- long double **std::modf** (long double \_\_x, long double \*\_\_iptr)
- constexpr float **std::pow** (float \_\_x, float \_\_y)
- constexpr long double **std::pow** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
constexpr \_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::pow** (\_Tp \_\_x, \_Up \_\_y)
- constexpr float **std::sin** (float \_\_x)
- constexpr long double **std::sin** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sin** (\_Tp \_\_x)
- constexpr float **std::sinh** (float \_\_x)
- constexpr long double **std::sinh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sinh** (\_Tp \_\_x)
- constexpr float **std::sqrt** (float \_\_x)
- constexpr long double **std::sqrt** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::sqrt** (\_Tp \_\_x)
- constexpr float **std::tan** (float \_\_x)
- constexpr long double **std::tan** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::tan** (\_Tp \_\_x)
- constexpr float **std::tanh** (float \_\_x)
- constexpr long double **std::tanh** (long double \_\_x)
- template<typename \_Tp >  
constexpr \_\_gnu\_cxx::\_\_enable\_if< \_\_is\_integer< \_Tp >::\_\_value, double >::\_\_type **std::tanh** (\_Tp \_\_x)

### 5.81.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.82 cmath File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_CMATH`

### 5.82.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.83 cmath File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CMATH`

## Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre` (unsigned int \_\_n, unsigned int \_\_m, \_Tp \_\_x)
- float `std::tr1::assoc_laguerref` (unsigned int \_\_n, unsigned int \_\_m, float \_\_x)
- long double `std::tr1::assoc_laguerrel` (unsigned int \_\_n, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre` (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_x)
- float `std::tr1::assoc_legendref` (unsigned int \_\_l, unsigned int \_\_m, float \_\_x)
- long double `std::tr1::assoc_legendrel` (unsigned int \_\_l, unsigned int \_\_m, long double \_\_x)
- `template<typename _Tpx, typename _Tpy >`  
`__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta` (\_Tpx \_\_x, \_Tpy \_\_y)
- float `std::tr1::betaf` (float \_\_x, float \_\_y)
- long double `std::tr1::betal` (long double \_\_x, long double \_\_y)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1` (\_Tp \_\_k)
- float `std::tr1::comp_ellint_1f` (float \_\_k)
- long double `std::tr1::comp_ellint_1l` (long double \_\_k)
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2` (\_Tp \_\_k)
- float `std::tr1::comp_ellint_2f` (float \_\_k)
- long double `std::tr1::comp_ellint_2l` (long double \_\_k)
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3` (\_Tp \_\_k, \_Tpn \_\_nu)
- float `std::tr1::comp_ellint_3f` (float \_\_k, float \_\_nu)
- long double `std::tr1::comp_ellint_3l` (long double \_\_k, long double \_\_nu)
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg` (\_Tpa \_\_a, \_Tpc \_\_c, \_Tp \_\_x)
- float `std::tr1::conf_hypergf` (float \_\_a, float \_\_c, float \_\_x)
- long double `std::tr1::conf_hypergl` (long double \_\_a, long double \_\_c, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::tr1::cyl_bessel_if` (float \_\_nu, float \_\_x)
- long double `std::tr1::cyl_bessel_il` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::tr1::cyl_bessel_jf` (float \_\_nu, float \_\_x)
- long double `std::tr1::cyl_bessel_jl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::tr1::cyl_bessel_kf` (float \_\_nu, float \_\_x)
- long double `std::tr1::cyl_bessel_kl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann` (\_Tpnu \_\_nu, \_Tp \_\_x)
- float `std::tr1::cyl_neumannf` (float \_\_nu, float \_\_x)
- long double `std::tr1::cyl_neumannl` (long double \_\_nu, long double \_\_x)
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1` (\_Tp \_\_k, \_Tpp \_\_phi)
- float `std::tr1::ellint_1f` (float \_\_k, float \_\_phi)
- long double `std::tr1::ellint_1l` (long double \_\_k, long double \_\_phi)
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2` (\_Tp \_\_k, \_Tpp \_\_phi)

- float **std::tr1::ellint\_2f** (float \_\_k, float \_\_phi)
- long double **std::tr1::ellint\_2l** (long double \_\_k, long double \_\_phi)
- template<typename \_Tp, typename \_Tpn, typename \_Tpp >  
\_\_gnu\_cxx::\_\_promote\_3< \_Tp, \_Tpn, \_Tpp >::\_\_type **std::tr1::ellint\_3** (\_Tp \_\_k, \_Tpn \_\_nu, \_Tpp \_\_phi)
- float **std::tr1::ellint\_3f** (float \_\_k, float \_\_nu, float \_\_phi)
- long double **std::tr1::ellint\_3l** (long double \_\_k, long double \_\_nu, long double \_\_phi)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::expint** (\_Tp \_\_x)
- float **std::tr1::expintf** (float \_\_x)
- long double **std::tr1::expintl** (long double \_\_x)
- float **std::tr1::fabs** (float \_\_x)
- long double **std::tr1::fabsl** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::fabs** (\_Tp \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::hermite** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::hermitef** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::hermitel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tpa, typename \_Tpb, typename \_Tpc, typename \_Tp >  
\_\_gnu\_cxx::\_\_promote\_4< \_Tpa, \_Tpb, \_Tpc, \_Tp >::\_\_type **std::tr1::hyperg** (\_Tpa \_\_a, \_Tpb \_\_b, \_Tpc \_\_c, \_Tp \_\_x)
- float **std::tr1::hypergf** (float \_\_a, float \_\_b, float \_\_c, float \_\_x)
- long double **std::tr1::hypergl** (long double \_\_a, long double \_\_b, long double \_\_c, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::laguerre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::laguerref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::laguerrel** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::legendre** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::legendref** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::legendrel** (unsigned int \_\_n, long double \_\_x)
- float **std::tr1::pow** (float \_\_x, float \_\_y)
- long double **std::tr1::powl** (long double \_\_x, long double \_\_y)
- template<typename \_Tp, typename \_Up >  
\_\_gnu\_cxx::\_\_promote\_2< \_Tp, \_Up >::\_\_type **std::tr1::pow** (\_Tp \_\_x, \_Up \_\_y)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::riemann\_zeta** (\_Tp \_\_x)
- float **std::tr1::riemann\_zetaf** (float \_\_x)
- long double **std::tr1::riemann\_zetal** (long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_bessel** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_besself** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_bessell** (unsigned int \_\_n, long double \_\_x)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_legendre** (unsigned int \_\_l, unsigned int \_\_m, \_Tp \_\_theta)
- float **std::tr1::sph\_legendref** (unsigned int \_\_l, unsigned int \_\_m, float \_\_theta)
- long double **std::tr1::sph\_legendrel** (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type **std::tr1::sph\_neumann** (unsigned int \_\_n, \_Tp \_\_x)
- float **std::tr1::sph\_neumannf** (unsigned int \_\_n, float \_\_x)
- long double **std::tr1::sph\_neumannl** (unsigned int \_\_n, long double \_\_x)



### 5.83.1 Detailed Description

This is a TR1 C++ Library header.

## 5.84 cmp\_fn\_imps.hpp File Reference

### 5.84.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s entire container comparison related functions.

## 5.85 codecvt File Reference

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX_CODECVT`
- `#define GLIBCXX_CODECVT_SPECIALIZATION(_NAME, _ELEM)`
- `#define GLIBCXX_CODECVT_SPECIALIZATION2(_NAME, _ELEM)`

### Enumerations

- enum `codecvt_mode` { `consume_header`, `generate_header`, `little_endian` }

### 5.85.1 Detailed Description

This is a Standard C++ Library header.

## 5.86 codecvt.h File Reference

### Classes

- class [std::\\_\\_codecvt\\_abstract\\_base<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT>](#)
- class [std::codecvt<char, char, mbstate\\_t>](#)
- class [std::codecvt<char16\\_t, char, mbstate\\_t>](#)
- class [std::codecvt<char32\\_t, char, mbstate\\_t>](#)
- class [std::codecvt<wchar\\_t, char, mbstate\\_t>](#)
- class [std::codecvt\\_base](#)
- class [std::codecvt\\_byname<\\_InternT, \\_ExternT, \\_StateT>](#)

## Namespaces

- [std](#)

### 5.86.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.87 `codecvt_specializations.h` File Reference

### Classes

- class [std::codecvt<\\_InternT, \\_ExternT, encoding\\_state>](#)
- struct [\\_\\_gnu\\_cxx::encoding\\_char\\_traits<\\_CharT>](#)
- class [\\_\\_gnu\\_cxx::encoding\\_state](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Functions

- `template<typename _Tp>  
size_t std::__iconv_adapter (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↵  
__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)`

### 5.87.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.88 `compare` File Reference

### 5.88.1 Detailed Description

This is a Standard C++ Library header.

## 5.89 `compatibility.h` File Reference

### 5.89.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

## 5.90 compatibility.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Tp >`  
`_Tp __gnu_parallel::__add_omp (volatile _Tp *__ptr, _Tp __addend)`
- `template<typename _Tp >`  
`bool __gnu_parallel::__cas_omp (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >`  
`bool __gnu_parallel::__compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)`
- `template<typename _Tp >`  
`_Tp __gnu_parallel::__fetch_and_add (volatile _Tp *__ptr, _Tp __addend)`
- `void __gnu_parallel::__yield ()`

#### 5.90.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.

This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

## 5.91 compiletime\_settings.h File Reference

### Macros

- `#define _GLIBCXX_CALL(__n)`
- `#define _GLIBCXX_PARALLEL_ASSERTIONS`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
- `#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
- `#define _GLIBCXX_SCALE_DOWN_FPU`
- `#define _GLIBCXX_VERBOSE_LEVEL`

#### 5.91.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

#### 5.91.2 Macro Definition Documentation

##### 5.91.2.1 \_GLIBCXX\_CALL

```
#define _GLIBCXX_CALL(
 __n)
```

Macro to produce log message when entering a function.

**Parameters**

|                        |             |
|------------------------|-------------|
| <code>_↔<br/>_n</code> | Input size. |
|------------------------|-------------|

**See also**

`_GLIBCXX_VERBOSE_LEVEL`

Definition at line 44 of file `completetime_settings.h`.

**5.91.2.2 `_GLIBCXX_PARALLEL_ASSERTIONS`**

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Should be switched on only locally.

Definition at line 61 of file `completetime_settings.h`.

**5.91.2.3 `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`**

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the L1 cache for `gnu_↔  
parallel::__parallel_random_shuffle()`.

Definition at line 68 of file `completetime_settings.h`.

**5.91.2.4 `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`**

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many `_GLIBCXX_PARALLEL_ASSERTIONS` in parallel code. Consider the size of the TLB for `gnu_parallel_↔  
::__parallel_random_shuffle()`.

Definition at line 74 of file `completetime_settings.h`.

### 5.91.2.5 \_GLIBCXX\_SCALE\_DOWN\_FPU

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

Definition at line 55 of file `completime_settings.h`.

### 5.91.2.6 \_GLIBCXX\_VERBOSE\_LEVEL

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

Definition at line 37 of file `completime_settings.h`.

## 5.92 complex File Reference

### Classes

- struct [std::complex< \\_Tp >](#)
- struct [std::complex< \\_Tp >](#)
- struct [std::complex< double >](#)
- struct [std::complex< float >](#)
- struct [std::complex< long double >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define __cpp_lib_complex_udls`
- `#define _GLIBCXX_COMPLEX`

## Functions

- `template<typename _Tp >`  
`_Tp std::__complex_abs (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::__complex_arg (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cos (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_cosh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_exp (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_log (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_pow_unsigned (complex< _Tp > __x, unsigned __n)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::__complex_proj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sin (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sinh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_sqrt (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tan (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`complex< _Tp > std::__complex_tanh (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::abs (const complex< _Tp > &)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acos (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::acosh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`_Tp std::arg (const complex< _Tp > &)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::arg (_Tp __x)`

- `template<typename _Tp >`  
`std::complex< _Tp > std::asin (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::asinh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atan (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::atanh (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::conj (const complex< _Tp > &)`
- `template<typename _Tp >`  
`constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::conj (_Tp __x)`
- `template<typename _Tp >`  
`complex< _Tp > std::cos (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::cosh (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::exp (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp std::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr _Tp std::imag (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::imag (_Tp)`
- `template<typename _Tp >`  
`complex< _Tp > std::log (const complex< _Tp > &)`
- `template<typename _Tp >`  
`complex< _Tp > std::log10 (const complex< _Tp > &)`
- `template<typename _Tp >`  
`_Tp constexpr std::norm (const complex< _Tp > &)`
- `template<typename _Tp >`  
`constexpr _Tp std::norm (const complex< _Tp > &__z)`
- `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::norm (_Tp __x)`
- `constexpr std::complex< double > std::literals::complex_literals::operator""i (long double __num)`
- `constexpr std::complex< double > std::literals::complex_literals::operator""i (unsigned long long __num)`
- `constexpr std::complex< float > std::literals::complex_literals::operator""if (long double __num)`
- `constexpr std::complex< float > std::literals::complex_literals::operator""if (unsigned long long __num)`
- `constexpr std::complex< long double > std::literals::complex_literals::operator""il (long double __num)`
- `constexpr std::complex< long double > std::literals::complex_literals::operator""il (unsigned long long __num)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)`
- `template<typename _Tp, typename _CharT, class _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)`
- `template<typename _Tp >`  
`complex< _Tp > std::polar (const _Tp &, const _Tp &=0)`

- `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, int)`
  - `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)`
  - `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp > &__x, const _Up &__y)`
  - `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const _Tp &__x, const std::complex< _Up > &__y)`
  - `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::pow (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)`
  - `template<typename _Tp >`  
`std::complex< _Tp > std::proj (const std::complex< _Tp > &)`
  - `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::proj (_Tp __x)`
  - `template<typename _Tp >`  
`constexpr _Tp std::real (const complex< _Tp > &__z)`
  - `template<typename _Tp >`  
`constexpr __gnu_cxx::__promote< _Tp >::__type std::real (_Tp __x)`
  - `template<typename _Tp >`  
`complex< _Tp > std::sin (const complex< _Tp > &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::sinh (const complex< _Tp > &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::sqrt (const complex< _Tp > &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::tan (const complex< _Tp > &)`
  - `template<typename _Tp >`  
`complex< _Tp > std::tanh (const complex< _Tp > &)`
- 
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)`
  - `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)`
- 
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)`



- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator\* (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator\* (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator\* (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator== (const _Tp &__x, const complex< _Tp > &__y)`
  
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const complex< _Tp > &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`constexpr bool std::operator!= (const _Tp &__x, const complex< _Tp > &__y)`

### 5.92.1 Detailed Description

This is a Standard C++ Library header.

## 5.93 complex File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_COMPLEX`

### Functions

- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::conj (const std::complex< _Tp > &__z)`
- `template<typename _Tp >`  
`std::complex< typename __gnu_cxx::__promote< _Tp >::__type > std::tr1::conj (_Tp __x)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::polar (const _Tp &__rho,`  
`const _Up &__theta)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const _Up &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const _Tp &__x,`  
`const std::complex< _Up > &__y)`
- `template<typename _Tp, typename _Up >`  
`std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex<`  
`_Tp > &__x, const std::complex< _Up > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const _Tp &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const _Tp &__x, const std::complex< _Tp > &__y)`
- `template<typename _Tp >`  
`std::complex< _Tp > std::tr1::pow (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)`

#### 5.93.1 Detailed Description

This is a TR1 C++ Library header.

## 5.94 complex.h File Reference

### Macros

- `#define _GLIBCXX_COMPLEX_H`

#### 5.94.1 Detailed Description

This is a Standard C++ Library header.

### 5.95 `concept_check.h` File Reference

#### Macros

- `#define __glibcxx_class_requires(_a, _b)`
- `#define __glibcxx_class_requires2(_a, _b, _c)`
- `#define __glibcxx_class_requires3(_a, _b, _c, _d)`
- `#define __glibcxx_class_requires4(_a, _b, _c, _d, _e)`
- `#define __glibcxx_function_requires(...)`

#### 5.95.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

### 5.96 `concepts` File Reference

#### Macros

- `#define _GLIBCXX_CONCEPTS`

#### 5.96.1 Detailed Description

This is a Standard C++ Library header.

### 5.97 `concurrency.h` File Reference

#### Classes

- class [`\_\_gnu\_cxx::\_\_scoped\_lock`](#)

#### Namespaces

- [`\_\_gnu\_cxx`](#)

#### Enumerations

- enum `_Lock_policy` { `_S_single`, `_S_mutex`, `_S_atomic` }

#### Functions

- void `__gnu_cxx::__throw_concurrency_lock_error()`
- void `__gnu_cxx::__throw_concurrency_unlock_error()`

#### Variables

- static const `_Lock_policy` `__gnu_cxx::__default_lock_policy`

#### 5.97.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.98 `cond_dealtor.hpp` File Reference

#### Classes

- class `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`

#### Namespaces

- `__gnu_pbds`

#### 5.98.1 Detailed Description

Contains a conditional deallocator.

### 5.99 `cond_key_dtor_entry_dealtor.hpp` File Reference

#### Classes

- class `__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >`

#### Namespaces

- `__gnu_pbds`

#### 5.99.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

## 5.100 condition\_variable File Reference

### Classes

- class [std::condition\\_variable](#)
- class [std::\\_V2::condition\\_variable\\_any](#)

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CONDITION_VARIABLE`

### Enumerations

- enum [std::cv\\_status](#) { `no_timeout`, `timeout` }

### Functions

- void **std::notify\_all\_at\_thread\_exit** (condition\_variable &, unique\_lock< mutex >)

#### 5.100.1 Detailed Description

This is a Standard C++ Library header.

## 5.101 const\_iterator.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_](#)< Value\_Type, Entry, Simple, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_BIN_HEAP_CIT_BASE`

#### 5.101.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

### 5.102 `const_iterator.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_const\\_iterator\\_< Node, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_BASIC_HEAP_CIT_BASE`
- `#define PB_DS_CLASS_C_DEC`

#### 5.102.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

### 5.103 `const_iterator.hpp` File Reference

#### 5.103.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.

This file is intended to be included inside a class definition, with `PB_DS_CLASS_C_DEC` defined to the name of the enclosing class.

### 5.104 `constructor_destructor_fn_imps.hpp` File Reference

#### 5.104.1 Detailed Description

Contains implementations of `cc_ht_map_`'s constructors, destructor, and related functions.

## 5.105 constructor\_destructor\_fn\_imps.hpp File Reference

### 5.105.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 5.106 constructor\_destructor\_fn\_imps.hpp File Reference

## 5.107 constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

### 5.107.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 5.108 constructor\_destructor\_no\_store\_hash\_fn\_imps.hpp File Reference

### 5.108.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 5.109 constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

### 5.109.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s constructors, destructor, and related functions.

## 5.110 constructor\_destructor\_store\_hash\_fn\_imps.hpp File Reference

### 5.110.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s constructors, destructor, and related functions.

## 5.111 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.111.1 Detailed Description

Contains an implementation class for binary\_heap\_.

## 5.112 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.112.1 Detailed Description

Contains an implementation for binomial\_heap\_.

## 5.113 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.113.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.114 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.114.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.115 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.115.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.116 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.116.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.117 constructors\_destructor\_fn\_imps.hpp File Reference

### 5.117.1 Detailed Description

Contains an implementation class for a pairing heap.



**5.118 constructors\_destructor\_fn\_imps.hpp File Reference****5.118.1 Detailed Description**

Contains an implementation class for pat\_trie.

**5.119 constructors\_destructor\_fn\_imps.hpp File Reference****5.119.1 Detailed Description**

Contains an implementation for rb\_tree\_.

**5.120 constructors\_destructor\_fn\_imps.hpp File Reference****5.120.1 Detailed Description**

Contains an implementation for rc\_binomial\_heap\_.

**5.121 constructors\_destructor\_fn\_imps.hpp File Reference****5.121.1 Detailed Description**

Contains an implementation class for splay\_tree\_.

**5.122 constructors\_destructor\_fn\_imps.hpp File Reference****5.122.1 Detailed Description**

Contains an implementation for thin\_heap\_.

**5.123 container\_base\_dispatch.hpp File Reference****Classes**

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, Mapped, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, cc\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, gp\\_hash\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, list\\_update\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, ov\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, pat\\_trie\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, rb\\_tree\\_tag, Policy\\_TI >](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch< Key, null\\_type, \\_Alloc, splay\\_tree\\_tag, Policy\\_TI >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_CHECK_KEY_DOES_NOT_EXIST(_Key)`
- `#define PB_DS_CHECK_KEY_EXISTS(_Key)`
- `#define PB_DS_DATA_FALSE_INDICATOR`
- `#define PB_DS_DATA_TRUE_INDICATOR`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_EP2VP(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2F(X)`
- `#define PB_DS_V2S(X)`
- `#define PB_DS_V2S(X)`

### 5.123.1 Detailed Description

Contains associative container dispatching.

## 5.124 `cpp_type_traits.h` File Reference

### Classes

- struct [std::iterator\\_traits<\\_Iterator>](#)

### Namespaces

- [std](#)

### Macros

- `#define __INT_N(TYPE)`

### Functions

- `template<typename _Iterator>`  
`constexpr _Iterator std::__miter_base (_Iterator __it)`

#### 5.124.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/type_traits>`.

### 5.125 `cpu_defines.h` File Reference

#### 5.125.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

### 5.126 `csetjmp` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSETJMP`
- `#define setjmp(env)`

#### 5.126.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 5.127 `csignal` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSIGNAL`

#### 5.127.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 5.128 `cstdalign` File Reference

#### Macros

- `#define _GLIBCXX_CSTDALIGN`

#### 5.128.1 Detailed Description

This is a Standard C++ Library header.

### 5.129 `cstdarg` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSTDARG`
- `#define va_end(ap)`

#### 5.129.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 5.130 `cstdarg` File Reference

#### Macros

- `#define _GLIBCXX_TR1_CSTDARG`

## 5.130.1 Detailed Description

This is a TR1 C++ Library header.

5.131 `cstdbool` File Reference

## Macros

- `#define _GLIBCXX_CSTDBOOL`

## 5.131.1 Detailed Description

This is a Standard C++ Library header.

5.132 `cstdbool` File Reference

## Macros

- `#define _GLIBCXX_TR1_CSTDBOOL`

## 5.132.1 Detailed Description

This is a TR1 C++ Library header.

5.133 `cstddef` File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_CSTDDEF`

## 5.133.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.134 cstdint File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTDINT`

### 5.134.1 Detailed Description

This is a Standard C++ Library header.

## 5.135 cstdint File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CSTDINT`

### 5.135.1 Detailed Description

This is a TR1 C++ Library header.

## 5.136 cstdio File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTDIO`

#### 5.136.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

### 5.137 **cstdio File Reference**

#### Macros

- `#define _GLIBCXX_TR1_CSTDIO`

#### 5.137.1 Detailed Description

This is a TR1 C++ Library header.

### 5.138 **cstdlib File Reference**

#### Namespaces

- [std](#)

#### Macros

- `#define _GLIBCXX_CSTDLIB`
- `#define EXIT_FAILURE`
- `#define EXIT_SUCCESS`

#### Functions

- void **std::abort** (void) throw ()
- int **std::atexit** (void(\*) (void)) throw ()
- void **std::exit** (int) throw ()

#### 5.138.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.139 cstdlib File Reference

### Macros

- `#define _GLIBCXX_TR1_CSTDLIB`

### 5.139.1 Detailed Description

This is a TR1 C++ Library header.

## 5.140 cstring File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CSTRING`

### Functions

- `void * std::memchr (void *__s, int __c, size_t __n)`
- `char * std::strchr (char *__s, int __n)`
- `char * std::strpbrk (char *__s1, const char *__s2)`
- `char * std::strrchr (char *__s, int __n)`
- `char * std::strstr (char *__s1, const char *__s2)`

### 5.140.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.141 ctgmath File Reference

### Macros

- `#define _GLIBCXX_CTGMATH`



## 5.141.1 Detailed Description

This is a Standard C++ Library header.

5.142 `ctgmath` File Reference

## Macros

- `#define _GLIBCXX_TR1_CTGMATH`

## 5.142.1 Detailed Description

This is a TR1 C++ Library header.

5.143 `ctime` File Reference

## Namespaces

- `std`

## Macros

- `#define _GLIBCXX_CTIME`

## 5.143.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

5.144 `ctime` File Reference

## Macros

- `#define _GLIBCXX_TR1_CTIME`

## 5.144.1 Detailed Description

This is a TR1 C++ Library header.

## 5.145 ctype\_base.h File Reference

### Classes

- struct [std::ctype\\_base](#)

### Namespaces

- [std](#)

#### 5.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.146 ctype\_inline.h File Reference

### Namespaces

- [std](#)

#### 5.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.147 cuchar File Reference

### Macros

- `#define _GLIBCXX_CUCHAR`

#### 5.147.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.148 `wchar` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CWCHAR`

### Functions

- `wchar_t * std::wcschr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wcspbrk` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wcsrchr` (`wchar_t *__p`, `wchar_t __c`)
- `wchar_t * std::wcsstr` (`wchar_t *__s1`, `const wchar_t *__s2`)
- `wchar_t * std::wmemchr` (`wchar_t *__p`, `wchar_t __c`, `size_t __n`)

#### 5.148.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.149 `wchar` File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCHAR`

#### 5.149.1 Detailed Description

This is a TR1 C++ Library header.

## 5.150 `cwctype` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CWCTYPE`

#### 5.150.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*.h` implementation files.

This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

## 5.151 `cwctype` File Reference

### Namespaces

- [std](#)
- [std::tr1](#)

### Macros

- `#define _GLIBCXX_TR1_CWCTYPE`

#### 5.151.1 Detailed Description

This is a TR1 C++ Library header.

## 5.152 `cxxabi.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::recursive\\_init\\_error](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [abi](#)

## Typedefs

- typedef \_\_cxa\_cdtor\_return\_type(\* \_\_cxxabiv1::\_\_cxa\_cdtor\_type) (void \*)

## Functions

- \_\_cxa\_dependent\_exception \* \_\_cxxabiv1::\_\_cxa\_allocate\_dependent\_exception () noexcept
- int \_\_cxxabiv1::\_\_cxa\_atexit (void\*)(void \*), void \*, void \*) noexcept
- void \_\_cxxabiv1::\_\_cxa\_bad\_cast ()
- void \_\_cxxabiv1::\_\_cxa\_bad\_typeid ()
- void \* \_\_cxxabiv1::\_\_cxa\_begin\_catch (void \*) noexcept
- std::type\_info \* \_\_cxxabiv1::\_\_cxa\_current\_exception\_type () noexcept
- void \_\_cxxabiv1::\_\_cxa\_deleted\_virtual (void)
- char \* \_\_cxxabiv1::\_\_cxa\_demangle (const char \* \_\_mangled\_name, char \* \_\_output\_buffer, size\_t \* \_\_length, int \* \_\_status)
- void \_\_cxxabiv1::\_\_cxa\_end\_catch ()
- int \_\_cxxabiv1::\_\_cxa\_finalize (void \*)
- void \_\_cxxabiv1::\_\_cxa\_free\_dependent\_exception (\_\_cxa\_dependent\_exception \*) noexcept
- void \_\_cxxabiv1::\_\_cxa\_free\_exception (void \*) noexcept
- void \* \_\_cxxabiv1::\_\_cxa\_get\_exception\_ptr (void \*) noexcept
- \_\_cxa\_eh\_globals \* \_\_cxxabiv1::\_\_cxa\_get\_globals () noexcept
- \_\_cxa\_eh\_globals \* \_\_cxxabiv1::\_\_cxa\_get\_globals\_fast () noexcept
- void \_\_cxxabiv1::\_\_cxa\_guard\_abort (\_\_guard \*) noexcept
- int \_\_cxxabiv1::\_\_cxa\_guard\_acquire (\_\_guard \*)
- void \_\_cxxabiv1::\_\_cxa\_guard\_release (\_\_guard \*) noexcept
- void \_\_cxxabiv1::\_\_cxa\_pure\_virtual (void)
- void \_\_cxxabiv1::\_\_cxa\_rethrow ()
- int \_\_cxxabiv1::\_\_cxa\_thread\_atexit (void\*)(void \*), void \*, void \*) noexcept
- void \_\_cxxabiv1::\_\_cxa\_throw (void \*, std::type\_info \*, void\*)(void \*)
- void \_\_cxxabiv1::\_\_cxa\_throw\_bad\_array\_new\_length ()
- \_\_cxa\_vec\_ctor\_return\_type \_\_cxxabiv1::\_\_cxa\_vec\_ctor (void \* \_\_dest\_array, void \* \_\_src\_array, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_cdtor\_return\_type(\* \_\_constructor)(void \*, void \*), \_\_cxa\_cdtor\_type \_\_destructor)
- void \_\_cxxabiv1::\_\_cxa\_vec\_cleanup (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_s, \_\_cxa\_cdtor\_type \_\_destructor) noexcept
- \_\_cxa\_vec\_ctor\_return\_type \_\_cxxabiv1::\_\_cxa\_vec\_ctor (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_cdtor\_type \_\_constructor, \_\_cxa\_cdtor\_type \_\_destructor)
- void \_\_cxxabiv1::\_\_cxa\_vec\_delete (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_destructor)
- void \_\_cxxabiv1::\_\_cxa\_vec\_delete2 (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_destructor, void(\* \_\_dealloc)(void \*))
- void \_\_cxxabiv1::\_\_cxa\_vec\_delete3 (void \* \_\_array\_address, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_destructor, void(\* \_\_dealloc)(void \*, size\_t))
- void \_\_cxxabiv1::\_\_cxa\_vec\_dtor (void \* \_\_array\_address, size\_t \_\_element\_count, size\_t \_\_element\_size, \_\_cxa\_cdtor\_type \_\_destructor)
- void \* \_\_cxxabiv1::\_\_cxa\_vec\_new (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_constructor, \_\_cxa\_cdtor\_type \_\_destructor)
- void \* \_\_cxxabiv1::\_\_cxa\_vec\_new2 (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_constructor, \_\_cxa\_cdtor\_type \_\_destructor, void(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void \*))
- void \* \_\_cxxabiv1::\_\_cxa\_vec\_new3 (size\_t \_\_element\_count, size\_t \_\_element\_size, size\_t \_\_padding\_size, \_\_cxa\_cdtor\_type \_\_constructor, \_\_cxa\_cdtor\_type \_\_destructor, void(\* \_\_alloc)(size\_t), void(\* \_\_dealloc)(void \*, size\_t))
- void \* \_\_cxxabiv1::\_\_dynamic\_cast (const void \* \_\_src\_ptr, const \_\_class\_type\_info \* \_\_src\_type, const \_\_class\_type\_info \* \_\_dst\_type, ptrdiff\_t \_\_src2dst)

### 5.152.1 Detailed Description

The header provides an interface to the C++ ABI.

### 5.152.2 Function Documentation

#### 5.152.2.1 `__cxa_demangle()`

```
char* __cxxabiv1::__cxa_demangle (
 const char * __mangled_name,
 char * __output_buffer,
 size_t * __length,
 int * __status)
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

#### Parameters

|                              |                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>__mangled_name</code>  | A NUL-terminated character string containing the name to be demangled.                                                                                                                                                                                                                                                                                                                                   |
| <code>__output_buffer</code> | A region of memory, allocated with <code>malloc</code> , of <code>*__length</code> bytes, into which the demangled name is stored. If <code>__output_buffer</code> is not long enough, it is expanded using <code>realloc</code> . <code>__output_buffer</code> may instead be <code>NULL</code> ; in that case, the demangled name is placed in a region of memory allocated with <code>malloc</code> . |
| <code>__length</code>        | If <code>__length</code> is non-null, the length of the buffer containing the demangled name is placed in <code>*__length</code> .                                                                                                                                                                                                                                                                       |
| <code>__status</code>        | If <code>__status</code> is non-null, <code>*__status</code> is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: <code>mangled_name</code> is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.                                                                                            |

#### Returns

A pointer to the start of the NUL-terminated demangled name, or `NULL` if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in `__gnu_cxx::__verbose_terminate_handler`.

See [https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext\\_demangling.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html) for other examples of use.

#### Note

The same demangling functionality is available via `libiberty` (`<libiberty/demangle.h>` and `libiberty`.↵a) in GCC 3.1 and later, but that requires explicit installation (`-enable-install-libiberty`) and uses a different API, although the ABI is unchanged.

## 5.153 `cxxabi_forced.h` File Reference

### Classes

- class [\\_\\_cxxabiv1::\\_\\_forced\\_unwind](#)

#### 5.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 5.154 `cxxabi_init_exception.h` File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_CDTOR_CALLABI`
- `#define _GLIBCXX_HAVE_CDTOR_CALLABI`

### Functions

- `void * __cxxabiv1::__cxa_allocate_exception (size_t) noexcept`
- `void __cxxabiv1::__cxa_free_exception (void *) noexcept`
- `__cxa_refcounted_exception * __cxxabiv1::__cxa_init_primary_exception (void *object, std::type\_info *tinfo, void(*dest)(void *)) noexcept`

#### 5.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.155 `cxxabi_tweaks.h` File Reference

### Macros

- `#define _GLIBCXX_CXA_VEC_CTOR_RETURN(x)`
- `#define _GLIBCXX_GUARD_BIT`
- `#define _GLIBCXX_GUARD_PENDING_BIT`
- `#define _GLIBCXX_GUARD_SET(x)`
- `#define _GLIBCXX_GUARD_TEST(x)`
- `#define _GLIBCXX_GUARD_WAITING_BIT`

### Typedefs

- typedef void `__cxxabiv1::__cxa_cdtor_return_type`
- typedef void `__cxxabiv1::__cxa_vec_ctor_return_type`

### Variables

- `__extension__` typedef int `__cxxabiv1::__guard`

### 5.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cxxabi.h>`.

## 5.156 debug.h File Reference

### Classes

- class [`\_\_gnu\_debug::\_Safe\_iterator<\_Iterator, \_Sequence, \_Category>`](#)

### Namespaces

- [`\_\_gnu\_debug`](#)
- [`std`](#)
- [`std::\_\_debug`](#)

### Macros

- `#define __glibcxx_requires_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_can_increment(_First, _Size)`
- `#define __glibcxx_requires_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_cond(_Cond, _Msg)`
- `#define __glibcxx_requires_heap(_First, _Last)`
- `#define __glibcxx_requires_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive(_First, _Last)`
- `#define __glibcxx_requires_irreflexive2(_First, _Last)`
- `#define __glibcxx_requires_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_requires_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_requires_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_requires_sorted(_First, _Last)`
- `#define __glibcxx_requires_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_requires_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_requires_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_requires_string(_String)`
- `#define __glibcxx_requires_string_len(_String, _Len)`
- `#define __glibcxx_requires_valid_range(_First, _Last)`



## 5.156.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.157 `debug_allocator.h` File Reference

## Classes

- class [\\_\\_gnu\\_cxx::debug\\_allocator<\\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.157.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.158 `debug_fn_imps.hpp` File Reference

## 5.158.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.159 `debug_fn_imps.hpp` File Reference

## 5.159.1 Detailed Description

Contains an implementation for `binomial_heap_`.

5.160 `debug_fn_imps.hpp` File Reference

## 5.160.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

5.161 `debug_fn_imps.hpp` File Reference

## 5.161.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.162 `debug_fn_imps.hpp` File Reference

### 5.162.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

## 5.163 `debug_fn_imps.hpp` File Reference

### 5.163.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

## 5.164 `debug_fn_imps.hpp` File Reference

### 5.164.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.165 `debug_fn_imps.hpp` File Reference

### 5.165.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

## 5.166 `debug_fn_imps.hpp` File Reference

### 5.166.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 5.167 `debug_fn_imps.hpp` File Reference

### 5.167.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.168 `debug_fn_imps.hpp` File Reference

### 5.168.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 5.169 `debug_fn_imps.hpp` File Reference

### 5.169.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 5.170 `debug_fn_imps.hpp` File Reference

### 5.170.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 5.171 `debug_fn_imps.hpp` File Reference

### 5.171.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 5.172 `debug_fn_imps.hpp` File Reference

### 5.172.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 5.173 `debug_map_base.hpp` File Reference

### 5.173.1 Detailed Description

Contains a debug-mode base for all maps.

### 5.174 `debug_no_store_hash_fn_imps.hpp` File Reference

#### 5.174.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

### 5.175 `debug_no_store_hash_fn_imps.hpp` File Reference

#### 5.175.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

### 5.176 `debug_store_hash_fn_imps.hpp` File Reference

#### 5.176.1 Detailed Description

Contains implementations of `cc_ht_map_`'s debug-mode functions.

### 5.177 `debug_store_hash_fn_imps.hpp` File Reference

#### 5.177.1 Detailed Description

Contains implementations of `gp_ht_map_`'s debug-mode functions.

### 5.178 `decimal` File Reference

#### Classes

- class [std::decimal::decimal128](#)
- class [std::decimal::decimal32](#)
- class [std::decimal::decimal64](#)

#### Namespaces

- [std](#)
- [std::decimal](#)

## Macros

- `#define _DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT(_Op)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_DEC(_Op, _T1, _T2, _T3)`
- `#define _DECLARE_DECIMAL_BINARY_OP_WITH_INT(_Op, _Tp)`
- `#define _DECLARE_DECIMAL_COMPARISON(_Op, _Tp)`
- `#define _GLIBCXX_DECIMAL`
- `#define _GLIBCXX_USE_DECIMAL_`

## Functions

- `double std::decimal::decimal128_to_double (decimal128 __d)`
- `float std::decimal::decimal128_to_float (decimal128 __d)`
- `long double std::decimal::decimal128_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal128_to_long_long (decimal128 __d)`
- `double std::decimal::decimal32_to_double (decimal32 __d)`
- `float std::decimal::decimal32_to_float (decimal32 __d)`
- `long double std::decimal::decimal32_to_long_double (decimal32 __d)`
- `long long std::decimal::decimal32_to_long_long (decimal32 __d)`
- `double std::decimal::decimal64_to_double (decimal64 __d)`
- `float std::decimal::decimal64_to_float (decimal64 __d)`
- `long double std::decimal::decimal64_to_long_double (decimal64 __d)`
- `long long std::decimal::decimal64_to_long_long (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal32 __d)`
- `double std::decimal::decimal_to_double (decimal64 __d)`
- `double std::decimal::decimal_to_double (decimal128 __d)`
- `float std::decimal::decimal_to_float (decimal32 __d)`
- `float std::decimal::decimal_to_float (decimal64 __d)`
- `float std::decimal::decimal_to_float (decimal128 __d)`
- `long double std::decimal::decimal_to_long_double (decimal32 __d)`
- `long double std::decimal::decimal_to_long_double (decimal64 __d)`
- `long double std::decimal::decimal_to_long_double (decimal128 __d)`
- `long long std::decimal::decimal_to_long_long (decimal32 __d)`
- `long long std::decimal::decimal_to_long_long (decimal64 __d)`
- `long long std::decimal::decimal_to_long_long (decimal128 __d)`
- `static decimal128 std::decimal::make_decimal128 (long long __coeff, int __exp)`
- `static decimal128 std::decimal::make_decimal128 (unsigned long long __coeff, int __exp)`
- `static decimal32 std::decimal::make_decimal32 (long long __coeff, int __exp)`
- `static decimal32 std::decimal::make_decimal32 (unsigned long long __coeff, int __exp)`
- `static decimal64 std::decimal::make_decimal64 (long long __coeff, int __exp)`
- `static decimal64 std::decimal::make_decimal64 (unsigned long long __coeff, int __exp)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator!= (decimal32 __lhs, unsigned long __rhs)`

- [illegible]

- decimal64 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator\*** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator\*** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator\*** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator+** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator+** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator+** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator+** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator+** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator+** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator+** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator+** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator-** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator-** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long \_\_rhs)



- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator-** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator-** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator-** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator-** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator-** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, long long \_\_rhs)
- decimal32 **std::decimal::operator/** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- decimal32 **std::decimal::operator/** (int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (long long \_\_lhs, decimal32 \_\_rhs)
- decimal32 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, long long \_\_rhs)
- decimal64 **std::decimal::operator/** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- decimal64 **std::decimal::operator/** (int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long \_\_lhs, decimal64 \_\_rhs)

- decimal64 **std::decimal::operator/** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (long long \_\_lhs, decimal64 \_\_rhs)
- decimal64 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long \_\_rhs)
- decimal128 **std::decimal::operator/** (long long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, long long \_\_rhs)
- decimal128 **std::decimal::operator/** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- decimal128 **std::decimal::operator/** (int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- decimal128 **std::decimal::operator/** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (unsigned long \_\_lhs, decimal128 \_\_rhs)

- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator<** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator<** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator<** (decimal128 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal32 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, int \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal32 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long \_\_rhs)
- bool **std::decimal::operator==** (decimal64 \_\_lhs, unsigned long long \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal64 \_\_rhs)
- bool **std::decimal::operator==** (int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned int \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (unsigned long \_\_lhs, decimal128 \_\_rhs)
- bool **std::decimal::operator==** (decimal128 \_\_lhs, decimal32 \_\_rhs)



- `bool std::decimal::operator> (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator> (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, int __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal32 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, long __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal64 __lhs, int __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, int __rhs)`
- `bool std::decimal::operator>= (int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long __rhs)`
- `bool std::decimal::operator>= (long long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal64 __rhs)`
- `bool std::decimal::operator>= (unsigned long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal32 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned int __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, long long __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned int __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (decimal128 __lhs, unsigned long long __rhs)`
- `bool std::decimal::operator>= (long __lhs, decimal128 __rhs)`
- `bool std::decimal::operator>= (unsigned long long __lhs, decimal128 __rhs)`

### 5.178.1 Detailed Description

This is a Standard C++ Library header.

## 5.179 deque File Reference

### Macros

- `#define _GLIBCXX_DEQUE`

### 5.179.1 Detailed Description

This is a Standard C++ Library header.

## 5.180 deque File Reference

### Classes

- class [std::\\_\\_debug::deque<\\_Tp, \\_Allocator >](#)
- class [std::\\_\\_debug::deque<\\_Tp, \\_Allocator >](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define _GLIBCXX_DEBUG_DEQUE`

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator!= (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator< (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator<= (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator== (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator> (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::__debug::operator>= (const deque<_Tp, _Alloc > &__lhs, const deque<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::__debug::swap (deque<_Tp, _Alloc > &__lhs, deque<_Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

## 5.180.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.181 deque File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_DEQUE`

## Typedefs

- `template<typename _Tp >  
using std::experimental::fundamentals_v2::pmr::deque = std::deque< _Tp, polymorphic_allocator< _Tp >  
>`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >  
void std::experimental::fundamentals_v2::erase (deque< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate >  
void std::experimental::fundamentals_v2::erase_if (deque< _Tp, _Alloc > &__cont, _Predicate __pred)`

## 5.181.1 Detailed Description

This is a TS C++ Library header.

## 5.182 deque.tcc File Reference

## Namespaces

- [std](#)

## Macros

- `#define _DEQUE_TCC`

## Functions

- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std:: copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr >`  
`__last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std:: copy_move_a1 (::Deque_iterator< _ITp, _IRef, _IPtr >`  
`__first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type std:: copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std:: copy_move_backward_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std:: copy_move_backward_a1 (::Deque_iterator< _ITp, _IRef, _IPtr >`  
`__first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * >`  
`> __result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type std:: copy_move_backward_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`__result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std:: copy_move_backward_dit (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std:: copy_move_dit (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr`  
`> __last, _OI __result)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std:: equal_aux1 (::Deque_iterator< _Tp, _Ref, _Ptr >`  
`__first1, ::Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2 >`  
`bool std:: equal_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1,`  
`_Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std:: equal_aux1 (_II __`  
`__first1, _II __last1, ::Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II >`  
`bool std:: equal_dit (const ::Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > &__last1, _II __first2)`
- `template<typename _Tp, typename _VTP >`  
`void std:: fill_a1 (const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque_iterator< _Tp, _Tp`  
`&, _Tp * > &__last, const _VTP &__value)`

## 5.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.



## 5.183 direct\_mask\_range\_hashing\_imp.hpp File Reference

### 5.183.1 Detailed Description

Contains a range-hashing policy implementation

## 5.184 direct\_mod\_range\_hashing\_imp.hpp File Reference

### 5.184.1 Detailed Description

Contains a range-hashing policy implementation

## 5.185 dynamic\_bitset File Reference

### Classes

- struct [std::tr2::\\_\\_dynamic\\_bitset\\_base< \\_WordT, \\_Alloc >](#)
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >](#)
- class [std::tr2::dynamic\\_bitset< \\_WordT, \\_Alloc >::reference](#)

### Namespaces

- [std](#)
- [std::tr2](#)

### Macros

- `#define GLIBCXX_TR2_DYNAMIC_BITSET`

### Functions

- [template<typename \\_CharT, typename \\_Traits, typename \\_WordT, typename \\_Alloc >  
std::basic\\_ostream< \\_CharT, \\_Traits > & std::tr2::operator<< \(std::basic\\_ostream< \\_CharT, \\_Traits > &\\_\\_os,  
const dynamic\\_bitset< \\_WordT, \\_Alloc > &\\_\\_x\)](#)
- [template<typename \\_WordT, typename \\_Alloc >  
bool std::tr2::operator!= \(const dynamic\\_bitset< \\_WordT, \\_Alloc > &\\_\\_lhs, const dynamic\\_bitset< \\_WordT, \\_Alloc  
> &\\_\\_rhs\)](#)
- [template<typename \\_WordT, typename \\_Alloc >  
bool std::tr2::operator<= \(const dynamic\\_bitset< \\_WordT, \\_Alloc > &\\_\\_lhs, const dynamic\\_bitset< \\_WordT, \\_Alloc  
> &\\_\\_rhs\)](#)

- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`
- `template<typename _WordT, typename _Alloc >`  
`dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)`

### 5.185.1 Detailed Description

This is a TR2 C++ Library header.

## 5.186 `dynamic_bitset.tcc` File Reference

### Namespaces

- [std](#)
- [std::tr2](#)

### Macros

- `#define \_GLIBCXX\_TR2\_DYNAMIC\_BITSET\_TCC`

### Functions

- `template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc >`  
`std::basic\_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic\_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)`

## 5.186.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<tr2/dynamic_bitset>`.

## 5.187 enable\_special\_members.h File Reference

## Classes

- struct [std::Enable\\_copy\\_move<\\_Copy, \\_CopyAssignment, \\_Move, \\_MoveAssignment, \\_Tag >](#)
- struct [std::Enable\\_default\\_constructor<\\_Switch, \\_Tag >](#)
- struct [std::Enable\\_destructor<\\_Switch, \\_Tag >](#)
- struct [std::Enable\\_special\\_members<\\_Default, \\_Destructor, \\_Copy, \\_CopyAssignment, \\_Move, \\_MoveAssignment, \\_Tag >](#)

## Namespaces

- [std](#)

## 5.187.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.188 enc\_filebuf.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::enc\\_filebuf<\\_CharT >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.188.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.189 entry\_cmp.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, No\\_Throw >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::entry\\_cmp<\\_VTp, Cmp\\_Fn, \\_Alloc, false >::type](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.189.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 5.190 `entry_list_fn_imps.hpp` File Reference

### 5.190.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entry-list related functions.

## 5.191 `entry_metadata_base.hpp` File Reference

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.191.1 Detailed Description

Contains an implementation for a list update map.

## 5.192 `entry_pred.hpp` File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred](#)< `_VTp`, `Pred`, `_Alloc`, `No_Throw` >
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred](#)< `_VTp`, `Pred`, `_Alloc`, `false` >
- struct [\\_\\_gnu\\_pbds::detail::entry\\_pred](#)< `_VTp`, `Pred`, `_Alloc`, `true` >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.192.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 5.193 eq\_by\_less.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::eq\\_by\\_less](#)< Key, Cmp\_Fn >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.193.1 Detailed Description

Contains an equivalence function.

## 5.194 equally\_split.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- template<typename \_DifferenceType, typename \_OutputIterator >  
\_OutputIterator [\\_\\_gnu\\_parallel::\\_\\_equally\\_split](#) (\_DifferenceType \_\_n, \_ThreadIndex \_\_num\_threads, \_OutputIterator \_\_s)
- template<typename \_DifferenceType >  
\_DifferenceType [\\_\\_gnu\\_parallel::\\_\\_equally\\_split\\_point](#) (\_DifferenceType \_\_n, \_ThreadIndex \_\_num\_threads, \_ThreadIndex \_\_thread\_no)

#### 5.194.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

## 5.195 erase\_fn\_imps.hpp File Reference

#### 5.195.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.196 `erase_fn_imps.hpp` File Reference

### 5.196.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.197 `erase_fn_imps.hpp` File Reference

### 5.197.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.198 `erase_fn_imps.hpp` File Reference

### 5.198.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions.

## 5.199 `erase_fn_imps.hpp` File Reference

### 5.199.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions.

## 5.200 `erase_fn_imps.hpp` File Reference

### 5.200.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.201 `erase_fn_imps.hpp` File Reference

### 5.201.1 Detailed Description

Contains implementations of `lu_map_`.

## 5.202 erase\_fn\_imps.hpp File Reference

### 5.202.1 Detailed Description

Contains an implementation class for ov\_tree\_.

## 5.203 erase\_fn\_imps.hpp File Reference

### 5.203.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.204 erase\_fn\_imps.hpp File Reference

### 5.204.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.205 erase\_fn\_imps.hpp File Reference

### 5.205.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.206 erase\_fn\_imps.hpp File Reference

### 5.206.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 5.207 erase\_fn\_imps.hpp File Reference

### 5.207.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 5.208 `erase_fn_imps.hpp` File Reference

### 5.208.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 5.209 `erase_if.h` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Functions

- `template<typename _Container, typename _Predicate >  
_Container::size_type std::__detail::__erase_nodes_if (_Container &__cont, _Predicate __pred)`

### 5.209.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.210 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.210.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is not stored.

## 5.211 `erase_no_store_hash_fn_imps.hpp` File Reference

### 5.211.1 Detailed Description

Contains implementations of `gp_ht_map_`'s erase related functions, when the hash value is not stored.

## 5.212 `erase_store_hash_fn_imps.hpp` File Reference

### 5.212.1 Detailed Description

Contains implementations of `cc_ht_map_`'s erase related functions, when the hash value is stored.



## 5.213 erase\_store\_hash\_fn\_imps.hpp File Reference

### 5.213.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s erase related functions, when the hash value is stored.

## 5.214 error\_constants.h File Reference

### Namespaces

- [std](#)

### Enumerations

- enum **errc** {  
    **address\_family\_not\_supported**, **address\_in\_use**, **address\_not\_available**, **already\_connected**,  
    **argument\_list\_too\_long**, **argument\_out\_of\_domain**, **bad\_address**, **bad\_file\_descriptor**,  
    **broken\_pipe**, **connection\_aborted**, **connection\_already\_in\_progress**, **connection\_refused**,  
    **connection\_reset**, **cross\_device\_link**, **destination\_address\_required**, **device\_or\_resource\_busy**,  
    **directory\_not\_empty**, **executable\_format\_error**, **file\_exists**, **file\_too\_large**,  
    **filename\_too\_long**, **function\_not\_supported**, **host\_unreachable**, **illegal\_byte\_sequence**,  
    **inappropriate\_io\_control\_operation**, **interrupted**, **invalid\_argument**, **invalid\_seek**,  
    **io\_error**, **is\_a\_directory**, **message\_size**, **network\_down**,  
    **network\_reset**, **network\_unreachable**, **no\_buffer\_space**, **no\_child\_process**,  
    **no\_lock\_available**, **no\_message**, **no\_protocol\_option**, **no\_space\_on\_device**,  
    **no\_such\_device\_or\_address**, **no\_such\_device**, **no\_such\_file\_or\_directory**, **no\_such\_process**,  
    **not\_a\_directory**, **not\_a\_socket**, **not\_connected**, **not\_enough\_memory**,  
    **operation\_in\_progress**, **operation\_not\_permitted**, **operation\_not\_supported**, **operation\_would\_block**,  
    **permission\_denied**, **protocol\_not\_supported**, **read\_only\_file\_system**, **resource\_deadlock\_would\_occur**,  
    **resource\_unavailable\_try\_again**, **result\_out\_of\_range**, **timed\_out**, **too\_many\_files\_open\_in\_system**,  
    **too\_many\_files\_open**, **too\_many\_links**, **too\_many\_symbolic\_link\_levels**, **wrong\_protocol\_type** }

### 5.214.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<system_error>`.

## 5.215 exception File Reference

### Classes

- class [std::bad\\_exception](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define __cpp_lib_uncaught_exceptions`
- `#define` `__EXCEPTION__`

## Typedefs

- `typedef void(* std::terminate\_handler) ()`
- `typedef void(* std::unexpected\_handler) ()`

## Functions

- `void \_\_gnu\_cxx::\_\_verbose\_terminate\_handler ()`
- `terminate_handler std::get\_terminate () noexcept`
- `unexpected_handler std::get\_unexpected () noexcept`
- `terminate_handler std::set\_terminate (terminate_handler) noexcept`
- `unexpected_handler std::set\_unexpected (unexpected_handler) noexcept`
- `void std::terminate () noexcept`
- `bool std::uncaught\_exception () noexcept`
- `int std::uncaught\_exceptions () noexcept`
- `void std::unexpected ()`

### 5.215.1 Detailed Description

This is a Standard C++ Library header.

## 5.216 `exception.h` File Reference

### Classes

- class [std::exception](#)

### Namespaces

- [std](#)

### 5.216.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.217 `exception.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::container\\_error](#)
- struct [\\_\\_gnu\\_pbds::insert\\_error](#)
- struct [\\_\\_gnu\\_pbds::join\\_error](#)
- struct [\\_\\_gnu\\_pbds::resize\\_error](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Functions

- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_container\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_insert\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_join\\_error\(\)](#)
- void [\\_\\_gnu\\_pbds::\\_\\_throw\\_resize\\_error\(\)](#)

#### 5.217.1 Detailed Description

Contains exception classes.

## 5.218 `exception_defines.h` File Reference

### Macros

- `#define __catch(X)`
- `#define __throw_exception_again`
- `#define __try`

#### 5.218.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 5.219 `exception_ptr.h` File Reference

### Classes

- class [std::\\_\\_exception\\_ptr::exception\\_ptr](#)

## Namespaces

- [std](#)

## Functions

- `exception_ptr` [std::current\\_exception](#) () noexcept
- `template<typename _Ex >`  
`exception_ptr` [std::make\\_exception\\_ptr](#) (\_Ex \_\_ex) noexcept
- `void` [std::rethrow\\_exception](#) (exception\_ptr)

### 5.219.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 5.220 extc++.h File Reference

### 5.220.1 Detailed Description

This is an implementation file for a precompiled header.

## 5.221 extptr\_allocator.h File Reference

## Classes

- `class` [\\_\\_gnu\\_cxx::\\_ExtPtr\\_allocator<\\_Tp>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Tp >`  
`void` [\\_\\_gnu\\_cxx::swap](#) (\_ExtPtr\_allocator<\_Tp> &\_\_larg, \_ExtPtr\_allocator<\_Tp> &\_\_rarg)

### 5.221.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## Author

Bob Walters

An example allocator which uses an alternative pointer type from `bits/pointer.h`. Supports test cases which confirm container support for alternative pointers.

## 5.222 features.h File Reference

### Macros

- `#define _GLIBCXX_BAL_QUICKSORT`
- `#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
- `#define _GLIBCXX_FIND_EQUAL_SPLIT`
- `#define _GLIBCXX_FIND_GROWING_BLOCKS`
- `#define _GLIBCXX_MERGESORT`
- `#define _GLIBCXX_QUICKSORT`
- `#define _GLIBCXX_TREE_DYNAMIC_BALANCING`
- `#define _GLIBCXX_TREE_FULL_COPY`
- `#define _GLIBCXX_TREE_INITIAL_SPLITTING`

### 5.222.1 Detailed Description

Defines on whether to include algorithm variants.

Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

### 5.222.2 Macro Definition Documentation

#### 5.222.2.1 \_GLIBCXX\_BAL\_QUICKSORT

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

#### See also

`__gnu_parallel::Settings::sort_algorithm`

Definition at line 55 of file features.h.

#### 5.222.2.2 \_GLIBCXX\_FIND\_CONSTANT\_SIZE\_BLOCKS

```
#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
```

Include the equal-sized blocks variant for `std::find`.

#### See also

`__gnu_parallel::Settings::find_algorithm`

Definition at line 67 of file features.h.

#### 5.222.2.3 `_GLIBCXX_FIND_EQUAL_SPLIT`

```
#define _GLIBCXX_FIND_EQUAL_SPLIT
```

Include the equal splitting variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 74 of file `features.h`.

#### 5.222.2.4 `_GLIBCXX_FIND_GROWING_BLOCKS`

```
#define _GLIBCXX_FIND_GROWING_BLOCKS
```

Include the growing blocks variant for `std::find`.

See also

`__gnu_parallel::_Settings::find_algorithm`

Definition at line 61 of file `features.h`.

#### 5.222.2.5 `_GLIBCXX_MERGESORT`

```
#define _GLIBCXX_MERGESORT
```

Include parallel multi-way mergesort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 41 of file `features.h`.

#### 5.222.2.6 `_GLIBCXX_QUICKSORT`

```
#define _GLIBCXX_QUICKSORT
```

Include parallel unbalanced quicksort.

See also

`__gnu_parallel::_Settings::sort_algorithm`

Definition at line 48 of file `features.h`.

#### 5.222.2.7 \_GLIBCXX\_TREE\_DYNAMIC\_BALANCING

```
#define _GLIBCXX_TREE_DYNAMIC_BALANCING
```

Include the dynamic balancing variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 91 of file `features.h`.

#### 5.222.2.8 \_GLIBCXX\_TREE\_FULL\_COPY

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of `_Rb_tree::insert_unique(_Iter beg, _Iter __end)` a full copy of the input elements is done.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 100 of file `features.h`.

#### 5.222.2.9 \_GLIBCXX\_TREE\_INITIAL\_SPLITTING

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for `_Rb_tree::insert_unique(_Iter beg, _Iter __end)`.

See also

`__gnu_parallel::_Rb_tree`

Definition at line 83 of file `features.h`.

### 5.223 fenv.h File Reference

#### 5.223.1 Detailed Description

This is a Standard C++ Library header.

## 5.224 filesystem File Reference

### Macros

- `#define _GLIBCXX_FILESYSTEM`

### 5.224.1 Detailed Description

This is a Standard C++ Library header.

## 5.225 filesystem File Reference

### Macros

- `#define __cpp_lib_experimental_filesystem`
- `#define _GLIBCXX_EXPERIMENTAL_FILESYSTEM`

### 5.225.1 Detailed Description

This is a TS C++ Library header.

## 5.226 find.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair<_RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair<_RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair<_RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector >`  
`std::pair<_RAIter1, _RAIter2 > \_\_gnu\_parallel::find\_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)`



#### 5.226.1 Detailed Description

Parallel implementation base for `std::find()`, `std::equal()` and related functions. This file is a GNU parallel extension to the Standard C++ Library.

### 5.227 find\_fn\_imps.hpp File Reference

#### 5.227.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 5.228 find\_fn\_imps.hpp File Reference

#### 5.228.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

### 5.229 find\_fn\_imps.hpp File Reference

#### 5.229.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 5.230 find\_fn\_imps.hpp File Reference

#### 5.230.1 Detailed Description

Contains implementations of `cc_ht_map_`'s find related functions.

### 5.231 find\_fn\_imps.hpp File Reference

#### 5.231.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions.

### 5.232 find\_fn\_imps.hpp File Reference

#### 5.232.1 Detailed Description

Contains implementations of `lu_map_`.

### 5.233 `find_fn_imps.hpp` File Reference

#### 5.233.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.234 `find_fn_imps.hpp` File Reference

#### 5.234.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 5.235 `find_fn_imps.hpp` File Reference

#### 5.235.1 Detailed Description

Contains an implementation for `rb_tree_`.

### 5.236 `find_fn_imps.hpp` File Reference

#### 5.236.1 Detailed Description

Contains an implementation class for `splay_tree_`.

### 5.237 `find_fn_imps.hpp` File Reference

#### 5.237.1 Detailed Description

Contains an implementation for `thin_heap_`.

### 5.238 `find_no_store_hash_fn_imps.hpp` File Reference

#### 5.238.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is not stored.

## 5.239 find\_selectors.h File Reference

## Classes

- [struct \\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector<\\_FIterator>](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#)
- [struct \\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## 5.239.1 Detailed Description

\_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

## 5.240 find\_store\_hash\_fn\_imps.hpp File Reference

## 5.240.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s find related functions, when the hash value is stored.

## 5.241 find\_store\_hash\_fn\_imps.hpp File Reference

## 5.241.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s insert related functions, when the hash value is stored.

## 5.242 for\_each.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- [template<typename \\_Iter, typename \\_UserOp, typename \\_Functionality, typename \\_Red, typename \\_Result> \\_UserOp \\_\\_gnu\\_parallel::\\_\\_for\\_each\\_template\\_random\\_access\(\\_Iter \\_\\_begin, \\_Iter \\_\\_end, \\_UserOp \\_\\_user←\\_op, \\_Functionality & \\_\\_functionality, \\_Red \\_\\_reduction, \\_Result \\_\\_reduction\\_start, \\_Result & \\_\\_output, typename \[std::iterator\\\_traits<\\\_Iter>::difference\\\_type\]\(#\) \\_\\_bound, \\_Parallelism \\_\\_parallelism\\_tag\)](#)

### 5.242.1 Detailed Description

Main interface for embarrassingly parallel functions.

The explicit implementation are in other header files, like `workstealing.h`, `par_loop.h`, `omp_loop.h`, and `omp_loop_↔static.h`. This file is a GNU parallel extension to the Standard C++ Library.

### 5.243 `for_each_selectors.h` File Reference

#### Classes

- struct `__gnu_parallel::__accumulate_binop_reduct<_BinOp>`
- struct `__gnu_parallel::__accumulate_selector<_It>`
- struct `__gnu_parallel::__adjacent_difference_selector<_It>`
- struct `__gnu_parallel::__count_if_selector<_It, _Diff>`
- struct `__gnu_parallel::__count_selector<_It, _Diff>`
- struct `__gnu_parallel::__fill_selector<_It>`
- struct `__gnu_parallel::__for_each_selector<_It>`
- struct `__gnu_parallel::__generate_selector<_It>`
- struct `__gnu_parallel::__generic_for_each_selector<_It>`
- struct `__gnu_parallel::__identity_selector<_It>`
- struct `__gnu_parallel::__inner_product_selector<_It, _It2, _Tp>`
- struct `__gnu_parallel::__max_element_reduct<_Compare, _It>`
- struct `__gnu_parallel::__min_element_reduct<_Compare, _It>`
- struct `__gnu_parallel::__replace_if_selector<_It, _Op, _Tp>`
- struct `__gnu_parallel::__replace_selector<_It, _Tp>`
- struct `__gnu_parallel::__transform1_selector<_It>`
- struct `__gnu_parallel::__transform2_selector<_It>`
- struct `__gnu_parallel::__DummyReduct`
- struct `__gnu_parallel::__Nothing`

#### Namespaces

- `__gnu_parallel`

### 5.243.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

### 5.244 `formatter.h` File Reference

#### Classes

- class `__gnu_debug::__Safe_iterator<_Iterator, _Sequence, _Category>`
- class `__gnu_debug::__Safe_local_iterator<_Iterator, _Sequence>`
- class `__gnu_debug::__Safe_sequence<_Sequence>`
- class `std::move_iterator<_Iterator>`
- class `std::reverse_iterator<_Iterator>`

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_debug](#)
- [std](#)

## Macros

- `#define __GLIBCXX_TYPEID(_Type)`

## Enumerations

- `enum _Debug_msg_id {`  
`__msg_valid_range, __msg_insert_singular, __msg_insert_different, __msg_erase_bad,`  
`__msg_erase_different, __msg_subscript_oob, __msg_empty, __msg_unpartitioned,`  
`__msg_unpartitioned_pred, __msg_unsorted, __msg_unsorted_pred, __msg_not_heap,`  
`__msg_not_heap_pred, __msg_bad_bitset_write, __msg_bad_bitset_read, __msg_bad_bitset_flip,`  
`__msg_self_splice, __msg_splice_alloc, __msg_splice_bad, __msg_splice_other,`  
`__msg_splice_overlap, __msg_init_singular, __msg_init_copy_singular, __msg_init_const_singular,`  
`__msg_copy_singular, __msg_bad_deref, __msg_bad_inc, __msg_bad_dec,`  
`__msg_iter_subscript_oob, __msg_advance_oob, __msg_retreat_oob, __msg_iter_compare_bad,`  
`__msg_compare_different, __msg_iter_order_bad, __msg_order_different, __msg_distance_bad,`  
`__msg_distance_different, __msg_deref_istream, __msg_inc_istream, __msg_output_ostream,`  
`__msg_deref_istreambuf, __msg_inc_istreambuf, __msg_insert_after_end, __msg_erase_after_bad,`  
`__msg_valid_range2, __msg_local_iter_compare_bad, __msg_non_empty_range, __msg_self_move_↵`  
`assign,`  
`__msg_bucket_index_oob, __msg_valid_load_factor, __msg_equal_allocs, __msg_insert_range_from_↵`  
`self,`  
`__msg_irreflexive_ordering }`

## Functions

- `template<typename _Iterator >`  
`bool __gnu_debug::__check_singular (const _Iterator &)`

## 5.244.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.245 forward\_list File Reference

## Macros

- `#define __GLIBCXX_FORWARD_LIST`

### 5.245.1 Detailed Description

This is a Standard C++ Library header.

## 5.246 forward\_list File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_Safe\\_forward\\_list<\\_SafeSequence >](#)
- class [std::\\_\\_debug::forward\\_list<\\_Tp, \\_Alloc >](#)
- class [std::\\_\\_debug::forward\\_list<\\_Tp, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define \_\_glibcxx\_check\_valid\_fl\_range(_First, _Last, _Dist)`
- `#define \_GLIBCXX20\_ONLY(__expr)`
- `#define \_GLIBCXX\_DEBUG\_FORWARD\_LIST`
- `#define \_GLIBCXX\_FWDLIST\_REMOVE\_RETURN\_TYPE\_TAG`

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator!= (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator< (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator<= (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator== (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator> (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator>= (const forward_list<_Tp, _Alloc > &__lx, const forward_list<_Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >  
void std::\_\_debug::swap (forward_list<_Tp, _Alloc > &__lx, forward_list<_Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))`

## 5.246.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.247 forward\_list File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_FORWARD_LIST`

## Typedefs

- `template<typename _Tp >`  
`using std::experimental::fundamentals_v2::pmr::forward_list = std::forward\_list< _Tp, polymorphic\_allocator< _Tp > >`

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
`void std::experimental::fundamentals_v2::erase (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)`
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)`

## 5.247.1 Detailed Description

This is a TS C++ Library header.

## 5.248 forward\_list.h File Reference

## Classes

- `struct std::\_Fwd\_list\_base< _Tp, _Alloc >`
- `struct std::\_Fwd\_list\_const\_iterator< _Tp >`
- `struct std::\_Fwd\_list\_iterator< _Tp >`
- `struct std::\_Fwd\_list\_node< _Tp >`
- `struct std::\_Fwd\_list\_node\_base`
- `class std::forward\_list< _Tp, _Alloc >`

## Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator!= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator< (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator<= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator> (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::operator>= (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`
- `template<typename _Tp, typename _Alloc >`  
`void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__↵  
lx.swap(__ly)))`

### 5.248.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 5.249 forward\_list.tcc File Reference

## Namespaces

- [std](#)

## Macros

- `#define _FORWARD_LIST_TCC`
- `#define _GLIBCXX20_ONLY(__expr)`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)`



### 5.249.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<forward_list>`.

## 5.250 fs\_dir.h File Reference

### 5.250.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 5.251 fs\_dir.h File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Functions

- directory\_iterator **std::experimental::filesystem::v1::begin** (directory\_iterator \_\_iter) noexcept
- recursive\_directory\_iterator **std::experimental::filesystem::v1::begin** (recursive\_directory\_iterator \_\_iter) noexcept
- directory\_iterator **std::experimental::filesystem::v1::end** (directory\_iterator) noexcept
- recursive\_directory\_iterator **std::experimental::filesystem::v1::end** (recursive\_directory\_iterator) noexcept
- bool **std::experimental::filesystem::v1::operator!=** (const directory\_iterator &\_\_lhs, const directory\_iterator &\_\_rhs)
- bool **std::experimental::filesystem::v1::operator!=** (const recursive\_directory\_iterator &\_\_lhs, const recursive\_directory\_iterator &\_\_rhs)
- bool **std::experimental::filesystem::v1::operator==** (const directory\_iterator &\_\_lhs, const directory\_iterator &\_\_rhs)
- bool **std::experimental::filesystem::v1::operator==** (const recursive\_directory\_iterator &\_\_lhs, const recursive\_directory\_iterator &\_\_rhs)

### 5.251.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 5.252 fs\_fwd.h File Reference

### 5.252.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 5.253 fs\_fwd.h File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Typedefs

- using **std::experimental::filesystem::v1::file\_time\_type** = std::chrono::system\_clock::time\_point

### Enumerations

- enum [std::experimental::filesystem::v1::copy\\_options](#) : unsigned short { **none**, **skip\_existing**, **overwrite\_existing**, **update\_existing**, **recursive**, **copy\_symlinks**, **skip\_symlinks**, **directories\_only**, **create\_symlinks**, **create\_hard\_links** }
- enum **directory\_options** : unsigned char { **none**, **follow\_directory\_symlink**, **skip\_permission\_denied** }
- enum **file\_type** : signed char { **none**, **not\_found**, **regular**, **directory**, **symlink**, **block**, **character**, **fifo**, **socket**, **unknown** }
- enum [std::experimental::filesystem::v1::perms](#) : unsigned { **none**, **owner\_read**, **owner\_write**, **owner\_exec**, **owner\_all**, **group\_read**, **group\_write**, **group\_exec**, **group\_all**, **others\_read**, **others\_write**, **others\_exec**, **others\_all**, **all**, **set\_uid**, **set\_gid**, **sticky\_bit**, **mask**, **unknown**, **add\_perms**, **remove\_perms**, **symlink\_nofollow** }

### Functions

- void **std::experimental::filesystem::v1::copy** (const path &\_\_from, const path &\_\_to, copy\_options \_\_options)
- void **std::experimental::filesystem::v1::copy** (const path &\_\_from, const path &\_\_to, copy\_options \_\_options, error\_code &) noexcept
- bool **std::experimental::filesystem::v1::copy\_file** (const path &\_\_from, const path &\_\_to, copy\_options \_\_↵ option)
- bool **std::experimental::filesystem::v1::copy\_file** (const path &\_\_from, const path &\_\_to, copy\_options \_\_↵ option, error\_code &) noexcept
- path **std::experimental::filesystem::v1::current\_path** ()
- bool **std::experimental::filesystem::v1::is\_regular\_file** (file\_status) noexcept
- bool **std::experimental::filesystem::v1::is\_symlink** (file\_status) noexcept
- constexpr copy\_options **std::experimental::filesystem::v1::operator&** (copy\_options \_\_x, copy\_options \_\_y) noexcept
- constexpr perms **std::experimental::filesystem::v1::operator&** (perms \_\_x, perms \_\_y) noexcept
- constexpr directory\_options **std::experimental::filesystem::v1::operator&** (directory\_options \_\_x, directory\_↵ options \_\_y) noexcept
- copy\_options & **std::experimental::filesystem::v1::operator&=** (copy\_options &\_\_x, copy\_options \_\_y) noexcept

- `perms & std::experimental::filesystem::v1::operator&= (perms __x, perms __y) noexcept`
- `directory_options & std::experimental::filesystem::v1::operator&= (directory_options __x, directory_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::v1::operator^ (copy_options __x, copy_options __y) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator^ (perms __x, perms __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::v1::operator^ (directory_options __x, directory_options __y) noexcept`
- `copy_options & std::experimental::filesystem::v1::operator^= (copy_options __x, copy_options __y) noexcept`
- `perms & std::experimental::filesystem::v1::operator^= (perms __x, perms __y) noexcept`
- `directory_options & std::experimental::filesystem::v1::operator^= (directory_options __x, directory_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::v1::operator| (copy_options __x, copy_options __y) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator| (perms __x, perms __y) noexcept`
- `constexpr directory_options std::experimental::filesystem::v1::operator| (directory_options __x, directory_options __y) noexcept`
- `copy_options & std::experimental::filesystem::v1::operator|= (copy_options __x, copy_options __y) noexcept`
- `perms & std::experimental::filesystem::v1::operator|= (perms __x, perms __y) noexcept`
- `directory_options & std::experimental::filesystem::v1::operator|= (directory_options __x, directory_options __y) noexcept`
- `constexpr copy_options std::experimental::filesystem::v1::operator~ (copy_options __x) noexcept`
- `constexpr perms std::experimental::filesystem::v1::operator~ (perms __x) noexcept`
- `constexpr directory_options std::experimental::filesystem::v1::operator~ (directory_options __x) noexcept`
- `file_status std::experimental::filesystem::v1::status (const path &)`
- `file_status std::experimental::filesystem::v1::status (const path &, error_code &) noexcept`
- `bool std::experimental::filesystem::v1::status_known (file_status) noexcept`
- `file_status std::experimental::filesystem::v1::symlink_status (const path &)`
- `file_status std::experimental::filesystem::v1::symlink_status (const path &, error_code &) noexcept`

#### 5.253.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 5.254 fs\_path.h File Reference

#### 5.254.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<filesystem>`.

## 5.255 fs\_path.h File Reference

### Classes

- class [std::experimental::filesystem::v1::filesystem\\_error](#)
- class [std::experimental::filesystem::v1::path::iterator](#)
- class [std::experimental::filesystem::v1::path](#)

### Namespaces

- [std](#)
- [std::experimental](#)

### 5.255.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/filesystem>`.

## 5.256 fstream File Reference

### Classes

- class [std::basic\\_filebuf<\\_CharT, \\_Traits>](#)
- class [std::basic\\_fstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ifstream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ofstream<\\_CharT, \\_Traits>](#)

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_FSTREAM`

### Functions

- `template<class _CharT, class _Traits>  
void std::swap (basic_filebuf<_CharT, _Traits> &__x, basic_filebuf<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_ifstream<_CharT, _Traits> &__x, basic_ifstream<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_ofstream<_CharT, _Traits> &__x, basic_ofstream<_CharT, _Traits> &__y)`
- `template<class _CharT, class _Traits>  
void std::swap (basic_fstream<_CharT, _Traits> &__x, basic_fstream<_CharT, _Traits> &__y)`

## 5.256.1 Detailed Description

This is a Standard C++ Library header.

5.257 **fstream.tcc** File Reference

## Namespaces

- [std](#)

## Macros

- `#define _FSTREAM_TCC`

## 5.257.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<fstream>`.

5.258 **functexcept.h** File Reference

## Namespaces

- [std](#)

## Functions

- void **std::\_\_throw\_bad\_alloc** (void)
- void **std::\_\_throw\_bad\_cast** (void)
- void **std::\_\_throw\_bad\_exception** (void)
- void **std::\_\_throw\_bad\_function\_call** ()
- void **std::\_\_throw\_bad\_typeid** (void)
- void **std::\_\_throw\_domain\_error** (const char \*)
- void **std::\_\_throw\_future\_error** (int)
- void **std::\_\_throw\_invalid\_argument** (const char \*)
- void **std::\_\_throw\_ios\_failure** (const char \*)
- void **std::\_\_throw\_ios\_failure** (const char \*, int)
- void **std::\_\_throw\_length\_error** (const char \*)
- void **std::\_\_throw\_logic\_error** (const char \*)
- void **std::\_\_throw\_out\_of\_range** (const char \*)
- void **std::\_\_throw\_out\_of\_range\_fmt** (const char \*,...)
- void **std::\_\_throw\_overflow\_error** (const char \*)
- void **std::\_\_throw\_range\_error** (const char \*)
- void **std::\_\_throw\_runtime\_error** (const char \*)
- void **std::\_\_throw\_system\_error** (int)
- void **std::\_\_throw\_underflow\_error** (const char \*)

### 5.258.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

This header provides support for `-fno-exceptions`.

## 5.259 functional File Reference

### Classes

- struct `std::_Bind<_Signature>`
- struct `std::_Bind_result<_Result, _Signature>`
- class `std::_Mu<_Arg, _IsBindExp, _IsPlaceholder>`
- class `std::_Mu<_Arg, false, false>`
- class `std::_Mu<_Arg, false, true>`
- class `std::_Mu<_Arg, true, false>`
- class `std::_Mu<reference_wrapper<_Tp>, false, false>`
- class `std::_Not_fn<_Fn>`
- struct `std::_Placeholder<_Num>`
- struct `std::is_bind_expression<_Tp>`
- struct `std::is_bind_expression<_Bind<_Signature>>`
- struct `std::is_bind_expression<_Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const _Bind<_Signature>>`
- struct `std::is_bind_expression<const _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<const volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<const volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_bind_expression<volatile _Bind<_Signature>>`
- struct `std::is_bind_expression<volatile _Bind_result<_Result, _Signature>>`
- struct `std::is_placeholder<_Tp>`
- struct `std::is_placeholder<_Placeholder<_Num>>`

### Namespaces

- `std`
- `std::placeholders`

### Macros

- `#define _GLIBCXX_DEPR_BIND`
- `#define _GLIBCXX_FUNCTIONAL`
- `#define _GLIBCXX_NOT_FN_CALL_OP(_QUALS)`

## Typedefs

- `template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>  
using std::__is_socketlike = __or_< is_integral<_Tp2>, is_enum<_Tp2> >`
- `template<std::size_t __i, typename _Tuple >  
using std::Safe_tuple_element_t = typename enable_if<(__i< tuple_size<_Tuple>::value), tuple_element<__i, _Tuple> >::type::type`

## Functions

- `template<std::size_t _Ind, typename... _Tp>  
auto std::__volget (volatile tuple<_Tp... > &__tuple) -> __tuple_element_t<_Ind, tuple<_Tp... >> volatile &`
- `template<std::size_t _Ind, typename... _Tp>  
auto std::__volget (const volatile tuple<_Tp... > &__tuple) -> __tuple_element_t<_Ind, tuple<_Tp... >> const volatile &`
- `template<typename _Func, typename... _BoundArgs>  
constexpr Bind_helper<__is_socketlike<_Func>::value, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Result, typename _Func, typename... _BoundArgs>  
constexpr Bindres_helper<_Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)`
- `template<typename _Tp, typename _Class >  
constexpr Mem_fn<_Tp _Class::* > std::mem_fn (_Tp _Class::* __pm) noexcept`

## Variables

- `const _Placeholder< 1 > std::placeholders::_1`
- `const _Placeholder< 10 > std::placeholders::_10`
- `const _Placeholder< 11 > std::placeholders::_11`
- `const _Placeholder< 12 > std::placeholders::_12`
- `const _Placeholder< 13 > std::placeholders::_13`
- `const _Placeholder< 14 > std::placeholders::_14`
- `const _Placeholder< 15 > std::placeholders::_15`
- `const _Placeholder< 16 > std::placeholders::_16`
- `const _Placeholder< 17 > std::placeholders::_17`
- `const _Placeholder< 18 > std::placeholders::_18`
- `const _Placeholder< 19 > std::placeholders::_19`
- `const _Placeholder< 2 > std::placeholders::_2`
- `const _Placeholder< 20 > std::placeholders::_20`
- `const _Placeholder< 21 > std::placeholders::_21`
- `const _Placeholder< 22 > std::placeholders::_22`
- `const _Placeholder< 23 > std::placeholders::_23`
- `const _Placeholder< 24 > std::placeholders::_24`
- `const _Placeholder< 25 > std::placeholders::_25`
- `const _Placeholder< 26 > std::placeholders::_26`
- `const _Placeholder< 27 > std::placeholders::_27`
- `const _Placeholder< 28 > std::placeholders::_28`
- `const _Placeholder< 29 > std::placeholders::_29`
- `const _Placeholder< 3 > std::placeholders::_3`
- `const _Placeholder< 4 > std::placeholders::_4`
- `const _Placeholder< 5 > std::placeholders::_5`
- `const _Placeholder< 6 > std::placeholders::_6`
- `const _Placeholder< 7 > std::placeholders::_7`
- `const _Placeholder< 8 > std::placeholders::_8`
- `const _Placeholder< 9 > std::placeholders::_9`

### 5.259.1 Detailed Description

This is a Standard C++ Library header.

## 5.260 functional File Reference

### Classes

- class [\\_\\_gnu\\_cxx::binary\\_compose<\\_Operation1, \\_Operation2, \\_Operation3 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_binary\\_fun<\\_Result, \\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_unary\\_fun<\\_Result, \\_Argument >](#)
- struct [\\_\\_gnu\\_cxx::constant\\_void\\_fun<\\_Result >](#)
- struct [\\_\\_gnu\\_cxx::project1st<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::project2nd<\\_Arg1, \\_Arg2 >](#)
- struct [\\_\\_gnu\\_cxx::select1st<\\_Pair >](#)
- struct [\\_\\_gnu\\_cxx::select2nd<\\_Pair >](#)
- class [\\_\\_gnu\\_cxx::subtractive\\_rng](#)
- class [\\_\\_gnu\\_cxx::unary\\_compose<\\_Operation1, \\_Operation2 >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define \_EXT\_FUNCTIONAL`

### Functions

- `template<class _Operation1, class _Operation2 >`  
`unary_compose<_Operation1, _Operation2 > \_\_gnu\_cxx::compose1 (const _Operation1 &__fn1, const _↵`  
`Operation2 &__fn2)`
- `template<class _Operation1, class _Operation2, class _Operation3 >`  
`binary_compose<_Operation1, _Operation2, _Operation3 > \_\_gnu\_cxx::compose2 (const _Operation1 &__fn1,`  
`const _Operation2 &__fn2, const _Operation3 &__fn3)`
- `template<class _Result >`  
`constant_void_fun<_Result > \_\_gnu\_cxx::constant0 (const _Result &__val)`
- `template<class _Result >`  
`constant_unary_fun<_Result, _Result > \_\_gnu\_cxx::constant1 (const _Result &__val)`
- `template<class _Result >`  
`constant_binary_fun<_Result, _Result, _Result > \_\_gnu\_cxx::constant2 (const _Result &__val)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::plus<_Tp >)`
- `template<class _Tp >`  
`_Tp \_\_gnu\_cxx::identity\_element (std::multiplies<_Tp >)`
- `template<class _Ret, class _Tp, class _Arg >`  
`std::mem\_fun1\_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1 (_Ret(_Tp::*_f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`std::const\_mem\_fun1\_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1 (_Ret(_Tp::*_f)(_Arg) const)`
- `template<class _Ret, class _Tp, class _Arg >`  
`std::mem\_fun1\_ref\_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1\_ref (_Ret(_Tp::*_f)(_Arg))`
- `template<class _Ret, class _Tp, class _Arg >`  
`std::const\_mem\_fun1\_ref\_t<_Ret, _Tp, _Arg > \_\_gnu\_cxx::mem\_fun1\_ref (_Ret(_Tp::*_f)(_Arg) const)`



## 5.260.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.261 functional File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_boyer_moore_searching`
- `#define __cpp_lib_experimental_not_fn`
- `#define _GLIBCXX_EXPERIMENTAL_FUNCTIONAL`

## Typedefs

- `template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>`  
using `std::experimental::fundamentals_v1::boyer_moore_base_t` = `std::conditional_t< std::is_byte_↔`  
`like< _Val, _Pred >::value, __boyer_moore_array_base< _Diff, 256, _Pred >, __boyer_moore_map_base<`  
`_Val, _Diff, _Hash, _Pred > >`

## Functions

- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↔`  
`Predicate = equal_to<>>`  
`boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate >` `std::experimental::fundamentals_v1::make_boyer_moore_↔`  
`( _RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↔`  
`Predicate = equal_to<>>`  
`boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate >` `std::experimental::fundamentals_v1::make_boyer_moore_searcher↔`  
`( _RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>`  
`default_searcher< _ForwardIterator, _BinaryPredicate >` `std::experimental::fundamentals_v1::make_default_searcher↔`  
`( _ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())`
- `template<typename _Fn >`  
`auto` `std::experimental::fundamentals_v2::not_fn` `( _Fn &&__fn)` `noexcept(std::is_nothrow_constructible<`  
`std::decay_t< _Fn >, _Fn && >::value)`

## Variables

- `template<typename _Tp >`  
`constexpr bool` `std::experimental::fundamentals_v1::is_bind_expression_v`
- `template<typename _Tp >`  
`constexpr int` `std::experimental::fundamentals_v1::is_placeholder_v`

### 5.261.1 Detailed Description

This is a TS C++ Library header.

### 5.261.2 Function Documentation

#### 5.261.2.1 `make_boyer_moore_horspool_searcher()`

```
template<typename _RAIter , typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↵
::value_type>, typename _BinaryPredicate = equal_to<>>>
boyer_moore_horspool_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals↵
_v1::make_boyer_moore_horspool_searcher (
 _RAIter __pat_first,
 _RAIter __pat_last,
 _Hash __hf = _Hash(),
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for `boyer_moore_horspool_searcher`.

Definition at line 303 of file `experimental/functional`.

#### 5.261.2.2 `make_boyer_moore_searcher()`

```
template<typename _RAIter , typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↵
::value_type>, typename _BinaryPredicate = equal_to<>>>
boyer_moore_searcher<_RAIter, _Hash, _BinaryPredicate> std::experimental::fundamentals_v1::make↵
_boyer_moore_searcher (
 _RAIter __pat_first,
 _RAIter __pat_last,
 _Hash __hf = _Hash(),
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for `boyer_moore_searcher`.

Definition at line 293 of file `experimental/functional`.

#### 5.261.2.3 `make_default_searcher()`

```
template<typename _ForwardIterator , typename _BinaryPredicate = std::equal_to<>>>
default_searcher<_ForwardIterator, _BinaryPredicate> std::experimental::fundamentals_v1::make_↵
default_searcher (
 _ForwardIterator __pat_first,
 _ForwardIterator __pat_last,
 _BinaryPredicate __pred = _BinaryPredicate()) [inline]
```

Generator function for `default_searcher`.

Definition at line 283 of file `experimental/functional`.

## 5.261.2.4 not\_fn()

```
template<typename _Fn >
auto std::experimental::fundamentals_v2::not_fn (
 _Fn && __fn) [inline], [noexcept]
```

[func.not\_fn] Function template not\_fn

Definition at line 372 of file experimental/functional.

## 5.261.3 Variable Documentation

## 5.261.3.1 is\_bind\_expression\_v

```
template<typename _Tp >
constexpr bool std::experimental::fundamentals_v1::is_bind_expression_v
```

Variable template for std::is\_bind\_expression.

Definition at line 61 of file experimental/functional.

## 5.261.3.2 is\_placeholder\_v

```
template<typename _Tp >
constexpr int std::experimental::fundamentals_v1::is_placeholder_v
```

Variable template for std::is\_placeholder.

Definition at line 65 of file experimental/functional.

## 5.262 functional\_hash.h File Reference

## Classes

- struct [std::hash< \\_Tp >](#)
- struct [std::hash< \\_Tp >](#)
- struct [std::hash< \\_Tp \\* >](#)
- struct [std::hash< bool >](#)
- struct [std::hash< char >](#)
- struct [std::hash< char16\\_t >](#)
- struct [std::hash< char32\\_t >](#)
- struct [std::hash< double >](#)
- struct [std::hash< float >](#)
- struct [std::hash< int >](#)
- struct [std::hash< long >](#)
- struct [std::hash< long double >](#)
- struct [std::hash< long long >](#)
- struct [std::hash< short >](#)
- struct [std::hash< signed char >](#)
- struct [std::hash< unsigned char >](#)
- struct [std::hash< unsigned int >](#)
- struct [std::hash< unsigned long >](#)
- struct [std::hash< unsigned long long >](#)
- struct [std::hash< unsigned short >](#)
- struct [std::hash< wchar\\_t >](#)

## Namespaces

- [std](#)

## Macros

- `#define _Cxx_hashtable_define_trivial_hash( _Tp)`

### 5.262.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

### 5.263 `functions.h` File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Functions

- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool __gnu_debug::__check_partitioned_lower ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`constexpr bool __gnu_debug::__check_partitioned_lower ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value, _Pred __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool __gnu_debug::__check_partitioned_upper ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value)`
- `template<typename _ForwardIterator, typename _Tp, typename _Pred >`  
`constexpr bool __gnu_debug::__check_partitioned_upper ( _ForwardIterator __first, _ForwardIterator __last,`  
`const _Tp &__value, _Pred __pred)`
- `template<typename _InputIterator >`  
`constexpr bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted (const _InputIterator &__first, const _InputIterator &__last, _↵`  
`Predicate __pred)`
- `template<typename _InputIterator >`  
`constexpr bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, std::input\_iterator\_tag)`
- `template<typename _ForwardIterator >`  
`constexpr bool __gnu_debug::__check_sorted_aux ( _ForwardIterator __first, _ForwardIterator __last, std::forward\_iterator\_tag)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted_aux (const _InputIterator &, const _InputIterator &, _Predicate,`  
`std::input\_iterator\_tag)`

- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted_aux (_ForwardIterator __first, _ForwardIterator __last, _↵`  
`Predicate __pred, std::forward\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__↵`  
`last, const _InputIterator2 &)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted_set (const _InputIterator1 &__first, const _InputIterator1 &__↵`  
`last, const _InputIterator2 &, _Predicate __pred)`
- `template<typename _InputIterator >`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &↵`  
`__last, std::true\_type)`
- `template<typename _InputIterator >`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, std::↵`  
`__false_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &__first, const _InputIterator &↵`  
`__last, _Predicate __pred, std::true\_type)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool __gnu_debug::__check_sorted_set_aux (const _InputIterator &, const _InputIterator &, _↵`  
`Predicate, std::false\_type)`
- `template<typename _InputIterator >`  
`_InputIterator __gnu_debug::__check_valid_range (const _InputIterator &__first, const _InputIterator &__last,`  
`const char *__file, unsigned int __line, const char *__function)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _↵`  
`InputIterator __other, _InputIterator __other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral >`  
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _↵`  
`Integral, _Integral, std::true\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`_InputIterator __other, _InputIterator __other_end, std::false\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const _Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const _Safe_iterator< _OtherIterator,`  
`_Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, type-`  
`name _OtherCategory >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const`  
`_Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator,`  
`_OtherSequence, _OtherCategory > &)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const _InputIterator &__other, const _InputIterator &__other_end)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const _InputIterator &__other, const _InputIterator &__other_end, std::true\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator >`  
`bool __gnu_debug::__foreign_iterator_aux3 (const _Safe_iterator< _Iterator, _Sequence, _Category > &,`  
`const _InputIterator &, const _InputIterator &, std::false\_type)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it,`  
`const typename _Sequence::value_type *__other)`

- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool __gnu_debug::__foreign_iterator_aux4 (const _Safe_iterator< _Iterator, _Sequence, _Category > &,...)`
- `template<typename _Iterator >`  
`constexpr bool __gnu_debug::__is_irreflexive (_Iterator __it)`
- `template<typename _Iterator, typename _Pred >`  
`constexpr bool __gnu_debug::__is_irreflexive_pred (_Iterator __it, _Pred __pred)`

### 5.263.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.264 future File Reference

### Classes

- class `std::__basic_future< _Res >`
- struct `std::__future_base`
- struct `std::__future_base::_Result< _Res >`
- struct `std::__future_base::_Result< _Res & >`
- struct `std::__future_base::_Result< void >`
- struct `std::__future_base::_Result_alloc< _Res, _Alloc >`
- struct `std::__future_base::_Result_base`
- class `std::future< _Res >`
- class `std::future< _Res >`
- class `std::future< _Res & >`
- class `std::future< void >`
- class `std::future_error`
- struct `std::is_error_code_enum< future_errc >`
- class `std::packaged_task< _Res(_ArgTypes...)>`
- class `std::promise< _Res >`
- class `std::promise< _Res >`
- class `std::promise< _Res & >`
- class `std::promise< void >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res >`
- class `std::shared_future< _Res & >`
- class `std::shared_future< void >`

### Namespaces

- `std`

### Macros

- `#define _GLIBCXX_FUTURE`

## Typedefs

- `template<typename _Fn, typename... _Args>`  
`using std::__async_result_of = typename __invoke_result< typename decay< _Fn >::type, typename decay< _Args >::type... >::type`

## Enumerations

- `enum std::future_errc { future_already_retrieved, promise_already_satisfied, no_state, broken_promise }`
- `enum std::future_status { ready, timeout, deferred }`
- `enum std::launch { async, deferred }`

## Functions

- `template<typename _Signature, typename _Fn, typename _Alloc = std::allocator<int>>>`  
`static shared_ptr< __future_base::_Task_state_base< _Signature > > std::__create_task_state (_Fn &&__fn, const _Alloc &__a= _Alloc())`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)`
- `template<typename _Fn, typename... _Args>`  
`future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)`
- `const error_category & std::future_category () noexcept`
- `error_code std::make_error_code (future_errc __errc) noexcept`
- `error_condition std::make_error_condition (future_errc __errc) noexcept`
- `constexpr launch std::operator& (launch __x, launch __y)`
- `launch & std::operator&= (launch &__x, launch __y)`
- `constexpr launch std::operator^ (launch __x, launch __y)`
- `launch & std::operator^= (launch &__x, launch __y)`
- `constexpr launch std::operator| (launch __x, launch __y)`
- `launch & std::operator|= (launch &__x, launch __y)`
- `constexpr launch std::operator~ (launch __x)`
- `template<typename _Res >`  
`void std::swap (promise< _Res > &__x, promise< _Res > &__y) noexcept`
- `template<typename _Res, typename... _ArgTypes>`  
`void std::swap (packaged_task< _Res(_ArgTypes...) > &__x, packaged_task< _Res(_ArgTypes...) > &__y) noexcept`

### 5.264.1 Detailed Description

This is a Standard C++ Library header.

## 5.265 gp\_ht\_map.hpp File Reference

### Classes

- `class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_`

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_GP_HASH_NAME`
- `#define PB_DS_GP_HASH_TRAITS_BASE`
- `#define PB_DS_HASH_EQ_FN_C_DEC`
- `#define PB_DS_RANGED_PROBE_FN_C_DEC`

#### 5.265.1 Detailed Description

Contains an implementation class for general probing hash.

### 5.266 `gslice.h` File Reference

#### Classes

- class [std::gslice](#)

#### Namespaces

- [std](#)

#### 5.266.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.267 `gslice_array.h` File Reference

#### Classes

- class [std::gslice\\_array<\\_Tp>](#)

#### Namespaces

- [std](#)



## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 5.267.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.268 hash\_bytes.h File Reference

## Namespaces

- [std](#)

## Functions

- `size_t std::_Fnv_hash_bytes (const void *__ptr, size_t __len, size_t __seed)`
- `size_t std::_Hash_bytes (const void *__ptr, size_t __len, size_t __seed)`

## 5.268.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.269 hash\_eq\_fn.hpp File Reference

## Classes

- [struct \\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, Store\\_Hash >](#)
- [struct \\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, false >](#)
- [struct \\_\\_gnu\\_pbds::detail::hash\\_eq\\_fn< Key, Eq\\_Fn, \\_Alloc, true >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.269.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

## 5.270 `hash_exponential_size_policy_imp.hpp` File Reference

### 5.270.1 Detailed Description

Contains a resize size policy implementation.

## 5.271 `hash_fun.h` File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `size_t __gnu_cxx::__stl_hash_string` (const char \* \_\_s)

### 5.271.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.272 `hash_load_check_resize_trigger_imp.hpp` File Reference

### Macros

- `#define PB_DS_ASSERT_VALID(X)`

### 5.272.1 Detailed Description

Contains a resize trigger implementation.

## 5.273 `hash_load_check_resize_trigger_size_base.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, Hold\\_Size >](#)
- class [\\_\\_gnu\\_pbds::detail::hash\\_load\\_check\\_resize\\_trigger\\_size\\_base< Size\\_Type, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.273.1 Detailed Description

Contains an base holding size for some resize policies.

## 5.274 hash\_map File Reference

## Classes

- class [\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define _BACKWARD_HASH_MAP`

## Functions

- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator!= (const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, const hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc >  
bool __gnu_cxx::operator== (const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap<_Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_map<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`
- `template<class _Key, class _Tp, class _HashFn, class _EqKey, class _Alloc >  
void __gnu_cxx::swap (hash_multimap<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm1, hash_multimap<_Key, _Tp, _HashFn, _EqKey, _Alloc > &__hm2)`

## 5.274.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.275 hash\_policy.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mask\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::direct\\_mod\\_range\\_hashing< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_exponential\\_size\\_policy< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_load\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::hash\\_prime\\_size\\_policy](#)
- class [\\_\\_gnu\\_pbds::hash\\_standard\\_resize\\_policy< Size\\_Policy, Trigger\\_Policy, External\\_Size\\_Access, Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::linear\\_probe\\_fn< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::quadratic\\_probe\\_fn< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_SIZE_BASE_C_DEC`

### Enumerations

- enum { `num_distinct_sizes_32_bit`, `num_distinct_sizes_64_bit`, `num_distinct_sizes` }

### Variables

- static const std::size\_t `__gnu_pbds::detail::g_a_sizes` [`num_distinct_sizes_64_bit`]

## 5.275.1 Detailed Description

Contains hash-related policies.

## 5.276 hash\_prime\_size\_policy\_imp.hpp File Reference

## 5.276.1 Detailed Description

Contains a resize size policy implementation.

## 5.277 hash\_set File Reference

## Classes

- class [\\_\\_gnu\\_cxx::hash\\_multiset< \\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::hash\\_set< \\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define _BACKWARD_HASH_SET`

## Functions

- `template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >`  
`bool __gnu_cxx::operator!= (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Value , class _HashFcn , class _EqualKey , class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >`  
`bool __gnu_cxx::operator== (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >`  
`void __gnu_cxx::swap (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`
- `template<class _Val , class _HashFcn , class _EqualKey , class _Alloc >`  
`void __gnu_cxx::swap (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)`

#### 5.277.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

### 5.278 `hash_standard_resize_policy_imp.hpp` File Reference

#### 5.278.1 Detailed Description

Contains a resize policy implementation.

### 5.279 `hashtable.h` File Reference

#### Classes

- class `std::_Hashtable<_Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits >`

#### Namespaces

- `std`

#### Typedefs

- `template<typename _Tp, typename _Hash >`  
using **`std::__cache_default`** = `__not_< __and_< __is_fast_hash< _Hash >, __is_nothrow_invocable< const _Hash &, const _Tp & > >>`

#### 5.279.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

### 5.280 `hashtable.h` File Reference

#### Namespaces

- `__gnu_cxx`

#### Enumerations

- enum { **`_S_num_primes`** }

## Functions

- unsigned long **gnu\_cxx::\_\_stl\_next\_prime** (unsigned long \_\_n)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool **gnu\_cxx::operator!=** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Ex, class \_Eq, class \_All >  
bool **gnu\_cxx::operator==** (const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht1, const hashtable< \_Val, \_Key, \_HF, \_Ex, \_Eq, \_All > &\_\_ht2)
- template<class \_Val, class \_Key, class \_HF, class \_Extract, class \_EqKey, class \_All >  
void **gnu\_cxx::swap** (hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht1, hashtable< \_Val, \_Key, \_HF, \_Extract, \_EqKey, \_All > &\_\_ht2)

## 5.280.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.281 hashtable\_policy.h File Reference

## Classes

- struct std::\_\_detail::Default\_ranged\_hash
- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >
- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, Default\_ranged\_hash, false >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, Default\_ranged\_hash, true >
- struct std::\_\_detail::Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >
- struct std::\_\_detail::Hash\_node< \_Value, \_Cache\_hash\_code >
- struct std::\_\_detail::Hash\_node< \_Value, false >
- struct std::\_\_detail::Hash\_node< \_Value, true >
- struct std::\_\_detail::Hash\_node\_base
- struct std::\_\_detail::Hash\_node\_value\_base< \_Value >
- class std::Hashtable< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >
- struct std::\_\_detail::Hashtable\_alloc< \_NodeAlloc >
- struct std::\_\_detail::Hashtable\_alloc< \_NodeAlloc >
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, false >
- struct std::\_\_detail::Hashtable\_ebo\_helper< \_Nm, \_Tp, true >
- struct std::\_\_detail::Hashtable\_traits< \_Cache\_hash\_code, \_Constant\_iterators, \_Unique\_keys >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Constant\_iterators >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >
- struct std::\_\_detail::Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >
- struct std::\_\_detail::Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >

- struct `std::__detail::Local_const_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __constant_iterators, __cache >`
- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache_hash_code >`
- struct `std::__detail::Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, true >`
- struct `std::__detail::Map_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, _Unique_k`
- struct `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false >`
- struct `std::__detail::Map_base< _Key, _Pair, _Alloc, _Select1st, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true >`
- struct `std::__detail::Mask_range_hashing`
- struct `std::__detail::Mod_range_hashing`
- struct `std::__detail::Node_const_iterator< _Value, __constant_iterators, __cache >`
- struct `std::__detail::Node_iterator< _Value, __constant_iterators, __cache >`
- struct `std::__detail::Node_iterator_base< _Value, _Cache_hash_code >`
- struct `std::__detail::Power2_rehash_policy`
- struct `std::__detail::Prime_rehash_policy`
- struct `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, typenar`
- struct `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, false_ty`
- struct `std::__detail::Rehash_base< _Key, _Value, _Alloc, _ExtractKey, _Equal, _H1, _H2, _Hash, _RehashPolicy, _Traits, true_ty`

## Namespaces

- `std`
- `std::__detail`

## Typedefs

- `template<typename _Policy >`  
using `std::__detail::__has_load_factor` = `typename _Policy::__has_load_factor`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash >`  
using `std::__detail::__hash_code_for_local_iter` = `_Hash_code_storage< _Hash_code_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, false > >`

## Functions

- `std::size_t std::__detail::__clp2 (std::size_t __n) noexcept`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw ( _Iterator __first, _Iterator __last, std::input_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw ( _Iterator __first, _Iterator __last, std::forward_iterator_tag)`
- `template<class _Iterator >`  
`std::iterator_traits< _Iterator >::difference_type std::__detail::__distance_fw ( _Iterator __first, _Iterator __last)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool std::__detail::operator!= (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator!= (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`
- `template<typename _Value, bool _Cache_hash_code>`  
`bool std::__detail::operator== (const _Node_iterator_base< _Value, _Cache_hash_code > &__x, const _Node_iterator_base< _Value, _Cache_hash_code > &__y) noexcept`
- `template<typename _Key, typename _Value, typename _ExtractKey, typename _H1, typename _H2, typename _Hash, bool __cache>`  
`bool std::__detail::operator== (const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__x, const _Local_iterator_base< _Key, _Value, _ExtractKey, _H1, _H2, _Hash, __cache > &__y)`



## 5.281.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>` or `<unordered_set>`.

## 5.282 helper\_functions.h File Reference

## Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator<\\_Iterator, \\_Sequence, \\_Category>](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator<\\_Iterator, \\_Sequence>](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Enumerations

- enum [\\_\\_gnu\\_debug::Distance\\_precision](#) {  
[\\_\\_dp\\_none](#), [\\_\\_dp\\_equality](#), [\\_\\_dp\\_sign](#), [\\_\\_dp\\_sign\\_max\\_size](#),  
[\\_\\_dp\\_exact](#) }

## Functions

- template<typename \_Iterator >  
constexpr \_Iterator [\\_\\_gnu\\_debug::\\_\\_base](#) (\_Iterator \_\_it)
- template<typename \_InputIterator, typename \_Size >  
constexpr bool [\\_\\_gnu\\_debug::\\_\\_can\\_advance](#) (\_InputIterator, \_Size)
- template<typename \_Iterator, typename \_Sequence, typename \_Category, typename \_Size >  
bool [\\_\\_gnu\\_debug::\\_\\_can\\_advance](#) (const \_\_Safe\_iterator<\_Iterator, \_Sequence, \_Category> &, \_Size)
- template<typename \_Iterator >  
bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular](#) (const \_Iterator &)
- template<typename \_Tp >  
bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular](#) (\_Tp \*const &\_\_ptr)
- bool [\\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const void \*)
- template<typename \_Iterator >  
constexpr \_Distance\_traits<\_Iterator>::\_\_type [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (\_Iterator \_\_lhs, \_Iterator \_\_rhs, [std::random\\_access\\_iterator\\_tag](#))
- template<typename \_Iterator >  
constexpr \_Distance\_traits<\_Iterator>::\_\_type [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (\_Iterator \_\_lhs, \_Iterator \_\_↔  
rhs, [std::input\\_iterator\\_tag](#))
- template<typename \_Iterator >  
constexpr \_Distance\_traits<\_Iterator>::\_\_type [\\_\\_gnu\\_debug::\\_\\_get\\_distance](#) (\_Iterator \_\_lhs, \_Iterator \_\_↔  
rhs)
- template<typename \_Iterator >  
\_Iterator [\\_\\_gnu\\_debug::\\_\\_unsafe](#) (\_Iterator \_\_it)

- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range (_InputIterator __first, _InputIterator __last, typename _Distance↵  
_traits< _InputIterator >::__type &__dist)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_↵  
iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_↵  
iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range (_InputIterator __first, _InputIterator __last)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_↵  
_iterator< _Iterator, _Sequence, _Category > &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_↵  
_iterator< _Iterator, _Sequence > &)`
- `template<typename _Integral >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_Integral, _Integral, std::__true_type)`
- `template<typename _Integral >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_Integral, _Integral, typename _Distance_traits< _Integral  
>::__type &__dist, std::__true_type)`
- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::input\_iterator\_tag)`
- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, std::__false_type)`
- `template<typename _InputIterator >`  
`constexpr bool \_\_gnu\_debug::\_\_valid\_range\_aux (_InputIterator __first, _InputIterator __last, typename _↵  
Distance_traits< _InputIterator >::__type &__dist, std::__false_type)`

### 5.282.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.283 `indirect_array.h` File Reference

#### Classes

- class [std::indirect\\_array< \\_Tp >](#)

#### Namespaces

- [std](#)

#### Macros

- `#define DEFINE\_VALARRAY\_OPERATOR(Op, _Name)`

#### 5.283.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.284 info\_fn\_imps.hpp File Reference

#### 5.284.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 5.285 info\_fn\_imps.hpp File Reference

#### 5.285.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

### 5.286 info\_fn\_imps.hpp File Reference

#### 5.286.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container info related functions.

### 5.287 info\_fn\_imps.hpp File Reference

#### 5.287.1 Detailed Description

Contains implementations of `gp_ht_map_`'s entire container info related functions.

### 5.288 info\_fn\_imps.hpp File Reference

#### 5.288.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

### 5.289 info\_fn\_imps.hpp File Reference

#### 5.289.1 Detailed Description

Contains implementations of `lu_map_`.

## 5.290 `info_fn_imps.hpp` File Reference

### 5.290.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 5.291 `info_fn_imps.hpp` File Reference

### 5.291.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 5.292 `info_fn_imps.hpp` File Reference

### 5.292.1 Detailed Description

Contains an implementation for `rb_tree_`.

## 5.293 `info_fn_imps.hpp` File Reference

### 5.293.1 Detailed Description

Contains an implementation.

## 5.294 `initializer_list` File Reference

### Classes

- class [std::initializer\\_list<\\_E>](#)

### Namespaces

- [std](#)

### 5.294.1 Detailed Description

This is a Standard C++ Library header.

## 5.295 insert\_fn\_imps.hpp File Reference

### 5.295.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.296 insert\_fn\_imps.hpp File Reference

### 5.296.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.297 insert\_fn\_imps.hpp File Reference

### 5.297.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.298 insert\_fn\_imps.hpp File Reference

### 5.298.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s insert related functions.

## 5.299 insert\_fn\_imps.hpp File Reference

### 5.299.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s insert related functions.

## 5.300 insert\_fn\_imps.hpp File Reference

### 5.300.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

### 5.301 `insert_fn_imps.hpp` File Reference

#### 5.301.1 Detailed Description

Contains implementations of `lu_map_`.

### 5.302 `insert_fn_imps.hpp` File Reference

#### 5.302.1 Detailed Description

Contains an implementation class for `ov_tree_`.

### 5.303 `insert_fn_imps.hpp` File Reference

#### 5.303.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.304 `insert_fn_imps.hpp` File Reference

#### 5.304.1 Detailed Description

Contains an implementation for `rb_tree_`.

### 5.305 `insert_fn_imps.hpp` File Reference

#### 5.305.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

### 5.306 `insert_fn_imps.hpp` File Reference

#### 5.306.1 Detailed Description

Contains an implementation class for `splay_tree_`.

## 5.307 `insert_fn_imps.hpp` File Reference

### 5.307.1 Detailed Description

Contains an implementation for `thin_heap_`.

## 5.308 `insert_join_fn_imps.hpp` File Reference

### 5.308.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 5.309 `insert_no_store_hash_fn_imps.hpp` File Reference

### 5.309.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is not stored.

## 5.310 `insert_no_store_hash_fn_imps.hpp` File Reference

### 5.310.1 Detailed Description

Contains implementations of `gp_ht_map_`'s insert related functions, when the hash value is not stored.

## 5.311 `insert_store_hash_fn_imps.hpp` File Reference

### 5.311.1 Detailed Description

Contains implementations of `cc_ht_map_`'s insert related functions, when the hash value is stored.

## 5.312 `insert_store_hash_fn_imps.hpp` File Reference

### 5.312.1 Detailed Description

Contains implementations of `gp_ht_map_`'s find related functions, when the hash value is stored.

## 5.313 `invoke.h` File Reference

Namespaces

- [std](#)

## Typedefs

- `template<typename _Res, typename _Callable, typename... _Args>`  
`using std::__can_invoke_as_nonvoid = __enable_if_t< __and< __not< is_void< _Res > >, is_↵`  
`convertible< typename __invoke_result< _Callable, _Args... >::type, _Res >::value, _Res >`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`using std::__can_invoke_as_void = __enable_if_t< __and< is_void< _Res >, __is_invocable< _Callable,`  
`_Args... >::value, _Res >`

## Functions

- `template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>`  
`constexpr _Up && std::__invfwd (typename remove_reference< _Tp >::type &__t) noexcept`
- `template<typename _Callable, typename... _Args>`  
`constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args)`  
`noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)`
- `template<typename _Res, typename _Fn, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_other, _Fn &&__f, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)`
- `template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>`  
`constexpr _Res std::__invoke_impl (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __↵`  
`args)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _MemPtr, typename _Tp >`  
`constexpr _Res std::__invoke_impl (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr __can_invoke_as_nonvoid< _Res, _Callable, _Args... > std::__invoke_r (_Callable &&__fn, _Args`  
`&&... __args)`
- `template<typename _Res, typename _Callable, typename... _Args>`  
`constexpr __can_invoke_as_void< _Res, _Callable, _Args... > std::__invoke_r (_Callable &&__fn, _Args &&...`  
`__args)`

### 5.313.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.314 iomanip File Reference

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_quoted_string_io`
- `#define _GLIBCXX_IOMANIP`



## Functions

- `template<typename _MoneyT >`  
`_Get_money< _MoneyT > std::get\_money (_MoneyT &__mon, bool __intl=false)`
- `template<typename _CharT >`  
`_Get_time< _CharT > std::get\_time (std::tm *__tmb, const _CharT *__fmt)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _↵`  
`Resetiosflags __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setbase`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setfill<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _↵`  
`Setprecision __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_↵`  
`money< _MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, _Put_time<`  
`_CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _↵`  
`CharT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setprecision`  
`__f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)`
- `template<typename _CharT, typename _Traits, typename _MoneyT >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_money<`  
`_MoneyT > __f)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, _Get_time<`  
`_CharT > __f)`
- `template<typename _MoneyT >`  
`_Put_money< _MoneyT > std::put\_money (const _MoneyT &__mon, bool __intl=false)`

- `template<typename _CharT >`  
`_Put_time< _CharT > std::put\_time (const std::tm * __tmb, const _CharT * __fmt)`
- `template<typename _CharT >`  
`auto std::quoted (const _CharT * __string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto std::quoted (const basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`auto std::quoted (basic_string< _CharT, _Traits, _Alloc > & __string, _CharT __delim=_CharT(""), _CharT __escape = _CharT("\\"))`
- `_Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)`
- `_Setbase std::setbase (int __base)`
- `template<typename _CharT >`  
`_Setfill< _CharT > std::setfill (_CharT __c)`
- `_Setiosflags std::setiosflags (ios_base::fmtflags __mask)`
- `_Setprecision std::setprecision (int __n)`
- `_Setw std::setw (int __n)`

#### 5.314.1 Detailed Description

This is a Standard C++ Library header.

### 5.315 ios File Reference

#### Macros

- `#define _GLIBCXX_IOS`

#### 5.315.1 Detailed Description

This is a Standard C++ Library header.

### 5.316 ios\_base.h File Reference

#### Classes

- class [std::ios\\_base::failure](#)
- class [std::ios\\_base](#)

#### Namespaces

- [std](#)

## Enumerations

- enum `_ios_Fmtflags` {  
`_S_boolalpha`, `_S_dec`, `_S_fixed`, `_S_hex`,  
`_S_internal`, `_S_left`, `_S_oct`, `_S_right`,  
`_S_scientific`, `_S_showbase`, `_S_showpoint`, `_S_showpos`,  
`_S_skipws`, `_S_unitbuf`, `_S_uppercase`, `_S_adjustfield`,  
`_S_basefield`, `_S_floatfield`, `_S_ios_fmtflags_end`, `_S_ios_fmtflags_max`,  
`_S_ios_fmtflags_min` }
- enum `_ios_iostate` {  
`_S_goodbit`, `_S_badbit`, `_S_eofbit`, `_S_failbit`,  
`_S_ios_iostate_end`, `_S_ios_iostate_max`, `_S_ios_iostate_min` }
- enum `_ios_Openmode` {  
`_S_app`, `_S_ate`, `_S_bin`, `_S_in`,  
`_S_out`, `_S_trunc`, `_S_ios_openmode_end`, `_S_ios_openmode_max`,  
`_S_ios_openmode_min` }
- enum `_ios_Seekdir` { `_S_beg`, `_S_cur`, `_S_end`, `_S_ios_seekdir_end` }
- enum `std::io_errc` { `stream` }

## Functions

- `ios_base & std::boolalpha (ios_base & __base)`
- `ios_base & std::dec (ios_base & __base)`
- `ios_base & std::defaultfloat (ios_base & __base)`
- `ios_base & std::fixed (ios_base & __base)`
- `ios_base & std::hex (ios_base & __base)`
- `ios_base & std::hexfloat (ios_base & __base)`
- `ios_base & std::internal (ios_base & __base)`
- `const error_category & std::iostream_category () noexcept`
- `ios_base & std::left (ios_base & __base)`
- `error_code std::make_error_code (io_errc __e) noexcept`
- `error_condition std::make_error_condition (io_errc __e) noexcept`
- `ios_base & std::noboolalpha (ios_base & __base)`
- `ios_base & std::noshowbase (ios_base & __base)`
- `ios_base & std::noshowpoint (ios_base & __base)`
- `ios_base & std::noshowpos (ios_base & __base)`
- `ios_base & std::noskipws (ios_base & __base)`
- `ios_base & std::nounitbuf (ios_base & __base)`
- `ios_base & std::nouppercase (ios_base & __base)`
- `ios_base & std::oct (ios_base & __base)`
- `constexpr _ios_Fmtflags std::operator& (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator& (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_iostate std::operator& (_ios_iostate __a, _ios_iostate __b)`
- `const _ios_Fmtflags & std::operator&= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator&= (_ios_Openmode & __a, _ios_Openmode __b)`
- `const _ios_iostate & std::operator&= (_ios_iostate & __a, _ios_iostate __b)`
- `constexpr _ios_Fmtflags std::operator^ (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator^ (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_iostate std::operator^ (_ios_iostate __a, _ios_iostate __b)`
- `const _ios_Fmtflags & std::operator^= (_ios_Fmtflags & __a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator^= (_ios_Openmode & __a, _ios_Openmode __b)`

- `const _ios_istate & std::operator^= (_ios_istate &__a, _ios_istate __b)`
- `constexpr _ios_Fmtflags std::operator| (_ios_Fmtflags __a, _ios_Fmtflags __b)`
- `constexpr _ios_Openmode std::operator| (_ios_Openmode __a, _ios_Openmode __b)`
- `constexpr _ios_istate std::operator| (_ios_istate __a, _ios_istate __b)`
- `const _ios_Fmtflags & std::operator|= (_ios_Fmtflags &__a, _ios_Fmtflags __b)`
- `const _ios_Openmode & std::operator|= (_ios_Openmode &__a, _ios_Openmode __b)`
- `const _ios_istate & std::operator|= (_ios_istate &__a, _ios_istate __b)`
- `constexpr _ios_Fmtflags std::operator~ (_ios_Fmtflags __a)`
- `constexpr _ios_Openmode std::operator~ (_ios_Openmode __a)`
- `constexpr _ios_istate std::operator~ (_ios_istate __a)`
- `ios_base & std::right (ios_base &__base)`
- `ios_base & std::scientific (ios_base &__base)`
- `ios_base & std::showbase (ios_base &__base)`
- `ios_base & std::showpoint (ios_base &__base)`
- `ios_base & std::showpos (ios_base &__base)`
- `ios_base & std::skipws (ios_base &__base)`
- `ios_base & std::unitbuf (ios_base &__base)`
- `ios_base & std::uppercase (ios_base &__base)`

#### 5.316.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ios>`.

### 5.317 iosfwd File Reference

#### Classes

- class `std::basic_filebuf< _CharT, _Traits >`
- class `std::basic_fstream< _CharT, _Traits >`
- class `std::basic_ifstream< _CharT, _Traits >`
- class `std::basic_ios< _CharT, _Traits >`
- class `std::basic_iostream< _CharT, _Traits >`
- class `std::basic_istream< _CharT, _Traits >`
- class `std::basic_istreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_ofstream< _CharT, _Traits >`
- class `std::basic_ostream< _CharT, _Traits >`
- class `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`
- class `std::basic_streambuf< _CharT, _Traits >`
- class `std::basic_stringbuf< _CharT, _Traits, _Alloc >`
- class `std::basic_stringstream< _CharT, _Traits, _Alloc >`
- class `std::istreambuf_iterator< _CharT, _Traits >`
- class `std::ostreambuf_iterator< _CharT, _Traits >`

#### Namespaces

- `std`

## Macros

- `#define _GLIBCXX_IOSFWD`

## Typedefs

- `typedef basic_filebuf< char > std::filebuf`
- `typedef basic_fstream< char > std::fstream`
- `typedef basic_ifstream< char > std::ifstream`
- `typedef basic_ios< char > std::ios`
- `typedef basic_istream< char > std::istream`
- `typedef basic_istreamstream< char > std::istreamstream`
- `typedef basic_ofstream< char > std::ofstream`
- `typedef basic_ostream< char > std::ostream`
- `typedef basic_ostreamstream< char > std::ostreamstream`
- `typedef basic_streambuf< char > std::streambuf`
- `typedef basic_stringbuf< char > std::stringbuf`
- `typedef basic_stringstream< char > std::stringstream`
- `typedef basic_filebuf< wchar_t > std::wfilebuf`
- `typedef basic_fstream< wchar_t > std::wfstream`
- `typedef basic_ifstream< wchar_t > std::wifstream`
- `typedef basic_ios< wchar_t > std::wios`
- `typedef basic_istream< wchar_t > std::wistream`
- `typedef basic_istreamstream< wchar_t > std::wistreamstream`
- `typedef basic_ofstream< wchar_t > std::wofstream`
- `typedef basic_ostream< wchar_t > std::wostream`
- `typedef basic_ostreamstream< wchar_t > std::wostreamstream`
- `typedef basic_streambuf< wchar_t > std::wstreambuf`
- `typedef basic_stringbuf< wchar_t > std::wstringbuf`
- `typedef basic_stringstream< wchar_t > std::wstringstream`

### 5.317.1 Detailed Description

This is a Standard C++ Library header.

## 5.318 iostream File Reference

### Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_IOSTREAM`

## Variables

- static `ios_base::Init` [std::\\_\\_ioinit](#)

## Standard Stream Objects

The `<iostream>` header declares the eight standard stream objects. For other declarations, see <http://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html> and the [I/O forward declarations](#)

They are required by default to cooperate with the global C library's `FILE` streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- `istream` [std::cin](#)
- `ostream` [std::cout](#)
- `ostream` [std::cerr](#)
- `ostream` [std::clog](#)
- `wistream` [std::wcin](#)
- `wostream` [std::wcout](#)
- `wostream` [std::wcerr](#)
- `wostream` [std::wclog](#)

## 5.318.1 Detailed Description

This is a Standard C++ Library header.

## 5.319 istream File Reference

### Classes

- class [std::basic\\_iostream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_istream<\\_CharT, \\_Traits>::sentry](#)

### Namespaces

- [std](#)

### Macros

- `#define` [\\_GLIBCXX\\_ISTREAM](#)

### Typedefs

- `template<typename _Tp>`  
using [std::\\_\\_do\\_is\\_convertible\\_to\\_basic\\_istream\\_impl](#) = `decltype(__is_convertible_to_basic_istream_↵  
test(declval<typename remove_reference<_Tp>::type * >()))`
- `template<typename _Istream>`  
using [std::\\_\\_rvalue\\_istream\\_type](#) = `typename __is_convertible_to_basic_istream<_Istream>::__istream_↵  
type`

## Functions

- `template<typename _Ch, typename _Up >`  
`basic_istream< _Ch, _Up > & std::__is_convertible_to_basic_istream_test (basic_istream< _Ch, _Up > *)`
- `template<typename _Istream, typename _Tp >`  
`enable_if< __and< __not< is_lvalue_reference< _Istream > >, __is_convertible_to_basic_istream< _Istream >, __is_extractable< __rvalue_istream_type< _Istream >, _Tp && >::value, __rvalue_istream_type< _Istream > >::type std::operator>> (_Istream && __is, _Tp && __x)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > & __is)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __in, _CharT & __c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > & __in, unsigned char & __c)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > & __in, signed char & __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __in, _CharT * __s)`
- `template<>`  
`basic_istream< char > & std::operator>> (basic_istream< char > & __in, char * __s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > & __in, unsigned char * __s)`
- `template<class _Traits >`  
`basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > & __in, signed char * __s)`

### 5.319.1 Detailed Description

This is a Standard C++ Library header.

## 5.320 istream.tcc File Reference

### Namespaces

- `std`

### Macros

- `#define _ISTREAM_TCC`

## Functions

- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)`
- `template<typename _CharT, typename _Traits >`  
`basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT *__s)`

### 5.320.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<istream>`.

## 5.321 iterator File Reference

### Macros

- `#define __cpp_lib_null_iterators`
- `#define _GLIBCXX_ITERATOR`

### 5.321.1 Detailed Description

This is a Standard C++ Library header.

## 5.322 iterator File Reference

### Namespaces

- [`\_\_gnu\_cxx`](#)

### Macros

- `#define _EXT_ITERATOR`



## Functions

- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input\_iterator\_tag)`
- `template<typename _RandomAccessIterator, typename _Distance >`  
`void \_\_gnu\_cxx::\_\_distance (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator, typename _Distance >`  
`void \_\_gnu\_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)`

### 5.322.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.323 iterator File Reference

### Classes

- class [std::experimental::fundamentals\\_v2::ostream\\_joiner](#)< [\\_DelimT](#), [\\_CharT](#), [\\_Traits](#) >

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define \_\_cpp\_lib\_experimental\_ostream\_joiner`
- `#define \_GLIBCXX\_EXPERIMENTAL\_ITERATOR`

## Functions

- `template<typename _CharT, typename _Traits, typename _DelimT >`  
`ostream_joiner< decay\_t< \_DelimT >, \_CharT, \_Traits > std::experimental::fundamentals\_v2::make\_ostream\_joiner`  
`(basic\_ostream< \_CharT, \_Traits > &__os, \_DelimT &&__delimiter)`

### 5.323.1 Detailed Description

This is a TS C++ Library header.

### 5.323.2 Function Documentation

#### 5.323.2.1 `make_ostream_joiner()`

```
template<typename _CharT , typename _Traits , typename _DelimT >
ostream_joiner<decay_t<_DelimT>, _CharT, _Traits> std::experimental::fundamentals_v2::make_ostream_joiner (
 basic_ostream< _CharT, _Traits > & __os,
 _DelimT && __delimiter) [inline]
```

Object generator for `ostream_joiner`.

Definition at line 104 of file `experimental/iterator`.

### 5.324 `iterator.h` File Reference

#### Classes

- class [\\_\\_gnu\\_parallel::\\_IteratorPair<\\_Iterator1, \\_Iterator2, \\_IteratorCategory >](#)
- class [\\_\\_gnu\\_parallel::\\_IteratorTriple<\\_Iterator1, \\_Iterator2, \\_Iterator3, \\_IteratorCategory >](#)

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 5.324.1 Detailed Description

Helper iterator classes for the `std::transform()` functions. This file is a GNU parallel extension to the Standard C++ Library.

### 5.325 `iterator.hpp` File Reference

#### 5.325.1 Detailed Description

Contains an `iterator_` class used for ranging over the elements of the table.

This file is intended to be included inside a class definition, with `PB_DS_CLASS_C_DEC` defined to the name of the enclosing class.

### 5.326 `iterator_concepts.h` File Reference

#### 5.326.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.327 iterator\_fn\_imps.hpp File Reference

### 5.327.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s iterators related functions, e.g., begin().

## 5.328 iterators\_fn\_imps.hpp File Reference

### 5.328.1 Detailed Description

Contains an implementation class for a binary\_heap.

## 5.329 iterators\_fn\_imps.hpp File Reference

### 5.329.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.330 iterators\_fn\_imps.hpp File Reference

### 5.330.1 Detailed Description

Contains implementations of cc\_ht\_map\_'s iterators related functions, e.g., begin().

## 5.331 iterators\_fn\_imps.hpp File Reference

### 5.331.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.332 iterators\_fn\_imps.hpp File Reference

### 5.332.1 Detailed Description

Contains implementations of lu\_map\_.

### 5.333 `iterators_fn_imps.hpp` File Reference

#### 5.333.1 Detailed Description

Contains an implementation class for `ov_tree_`.

### 5.334 `iterators_fn_imps.hpp` File Reference

#### 5.334.1 Detailed Description

Contains an implementation class for `pat_trie`.

### 5.335 `left_child_next_sibling_heap_.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap< Value\\_Type, Cmp\\_Fn, Node\\_Metadata, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

#### 5.335.1 Detailed Description

Contains an implementation class for a basic heap.

### 5.336 `lfts_config.h` File Reference

#### 5.336.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

## 5.337 limits File Reference

## Classes

- struct [std::\\_\\_numeric\\_limits\\_base](#)
- struct [std::numeric\\_limits< \\_Tp >](#)
- struct [std::numeric\\_limits< bool >](#)
- struct [std::numeric\\_limits< char >](#)
- struct [std::numeric\\_limits< char16\\_t >](#)
- struct [std::numeric\\_limits< char32\\_t >](#)
- struct [std::numeric\\_limits< double >](#)
- struct [std::numeric\\_limits< float >](#)
- struct [std::numeric\\_limits< int >](#)
- struct [std::numeric\\_limits< long >](#)
- struct [std::numeric\\_limits< long double >](#)
- struct [std::numeric\\_limits< long long >](#)
- struct [std::numeric\\_limits< short >](#)
- struct [std::numeric\\_limits< signed char >](#)
- struct [std::numeric\\_limits< unsigned char >](#)
- struct [std::numeric\\_limits< unsigned int >](#)
- struct [std::numeric\\_limits< unsigned long >](#)
- struct [std::numeric\\_limits< unsigned long long >](#)
- struct [std::numeric\\_limits< unsigned short >](#)
- struct [std::numeric\\_limits< wchar\\_t >](#)

## Namespaces

- [std](#)

## Macros

- `#define \_\_glibcxx\_digits\(T\)`
- `#define \_\_glibcxx\_digits10\(T\)`
- `#define \_\_glibcxx\_digits10\_b\(T, B\)`
- `#define \_\_glibcxx\_digits\_b\(T, B\)`
- `#define \_\_glibcxx\_double\_has\_denorm\_loss`
- `#define \_\_glibcxx\_double\_tinyness\_before`
- `#define \_\_glibcxx\_double\_traps`
- `#define \_\_glibcxx\_float\_has\_denorm\_loss`
- `#define \_\_glibcxx\_float\_tinyness\_before`
- `#define \_\_glibcxx\_float\_traps`
- `#define \_\_glibcxx\_integral\_traps`
- `#define \_\_glibcxx\_long\_double\_has\_denorm\_loss`
- `#define \_\_glibcxx\_long\_double\_tinyness\_before`
- `#define \_\_glibcxx\_long\_double\_traps`
- `#define \_\_glibcxx\_max\(T\)`
- `#define \_\_glibcxx\_max\_b\(T, B\)`
- `#define \_\_glibcxx\_max\_digits10\(T\)`
- `#define \_\_glibcxx\_min\(T\)`

- `#define __glibcxx_min_b(T, B)`
- `#define __glibcxx_signed(T)`
- `#define __glibcxx_signed_b(T, B)`
- `#define __INT_N(TYPE, BITSIZE, EXT, UEXT)`
- `#define __INT_N_201103(TYPE)`
- `#define __INT_N_U201103(TYPE)`
- `#define _GLIBCXX_NUMERIC_LIMITS`

#### Enumerations

- enum `std::float_denorm_style` { `std::denorm_indeterminate`, `std::denorm_absent`, `std::denorm_present` }
- enum `std::float_round_style` { `round_indeterminate`, `std::round_toward_zero`, `std::round_to_nearest`, `std::round_toward_infinity`, `std::round_toward_neg_infinity` }

#### 5.337.1 Detailed Description

This is a Standard C++ Library header.

### 5.338 `linear_probe_fn_imp.hpp` File Reference

#### 5.338.1 Detailed Description

Contains a probe policy implementation

### 5.339 `list` File Reference

#### Macros

- `#define _GLIBCXX_LIST`

#### 5.339.1 Detailed Description

This is a Standard C++ Library header.

### 5.340 `list` File Reference

#### Classes

- class `std::__debug::list<_Tp, _Allocator>`
- class `std::__debug::list<_Tp, _Allocator>`

## Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX20_ONLY(__expr)`
- `#define _GLIBCXX_DEBUG_LIST`
- `#define _GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG`

## Functions

- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator!= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator< (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator<= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator== (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator> (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`bool std::__debug::operator>= (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >`  
`void std::__debug::swap (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)`

## 5.340.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.341 list File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_LIST`

## Typedefs

- `template<typename _Tp >`  
using **`std::experimental::fundamentals_v2::pmr::list`** = `std::list`<\_Tp, `polymorphic_allocator`<\_Tp> >

## Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
void **`std::experimental::fundamentals_v2::erase`** (`list`<\_Tp, \_Alloc> &\_\_cont, const \_Up &\_\_value)
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (`list`<\_Tp, \_Alloc> &\_\_cont, \_Predicate \_\_pred)

### 5.341.1 Detailed Description

This is a TS C++ Library header.

## 5.342 list.tcc File Reference

### Namespaces

- [`std`](#)

### Macros

- `#define` **`_GLIBCXX20_ONLY`**(\_\_expr)
- `#define` **`_LIST_TCC`**

### 5.342.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 5.343 list\_partition.h File Reference

### Namespaces

- [`\_\_gnu\_parallel`](#)



## Functions

- `template<typename _Iter >`  
`void \_\_gnu\_parallel::\_\_shrink (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t &__range_↔  
length)`
- `template<typename _Iter >`  
`void \_\_gnu\_parallel::\_\_shrink\_and\_double (std::vector< _Iter > &__os_starts, size_t &__count_to_two, size_t  
&__range_length, const bool __make_twice)`
- `template<typename _Iter, typename _FunctorType >`  
`size_t \_\_gnu\_parallel::list\_partition (const _Iter __begin, const _Iter __end, _Iter * __starts, size_t * __lengths,  
const int __num_parts, _FunctorType & __f, int __oversampling=0)`

### 5.343.1 Detailed Description

\_\_Functionality to split \_\_sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 5.344 list\_update\_policy.hpp File Reference

## Classes

- class `\_\_gnu\_pbds::lu\_counter\_policy< Max_Count, _Alloc >`
- class `\_\_gnu\_pbds::lu\_move\_to\_front\_policy< _Alloc >`

## Namespaces

- `\_\_gnu\_pbds`

### 5.344.1 Detailed Description

Contains policies for list update containers.

## 5.345 locale File Reference

## Macros

- `#define \_GLIBCXX\_LOCALE`

### 5.345.1 Detailed Description

This is a Standard C++ Library header.

## 5.346 locale\_classes.h File Reference

### Classes

- class [std::collate<\\_CharT>](#)
- class [std::collate\\_byname<\\_CharT>](#)
- class [std::locale::facet](#)
- class [std::locale::id](#)
- class [std::locale](#)

### Namespaces

- [std](#)

#### 5.346.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.347 locale\_classes.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define \_LOCALE\_CLASSES\_TCC`

### Functions

- `template<typename _Facet>  
bool std::has\_facet(const locale &__loc) throw ()`
- `template<typename _Facet>  
const _Facet & std::use\_facet(const locale &__loc)`

#### 5.347.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.348 locale\_conv.h File Reference

## Classes

- class [std::wbuffer\\_convert<\\_Codecvt, \\_Elem, \\_Tr>](#)
- class [std::wstring\\_convert<\\_Codecvt, \\_Elem, \\_Wide\\_alloc, \\_Byte\\_alloc>](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>  
bool std::__do_str_codecvt (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string<_CharT, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_in (const char *__first, const char *__last, basic_string<_CharT, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_in_all (const char *__first, const char *__last, basic_string<_CharT, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string<char, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt, _State &__state, size_t &__count)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_out (const _CharT *__first, const _CharT *__last, basic_string<char, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _State>  
bool std::__str_codecvt_out_all (const _CharT *__first, const _CharT *__last, basic_string<char, _Traits, _Alloc> &__outstr, const codecvt<_CharT, char, _State> &__cvt)`

## 5.348.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.349 locale\_facets.h File Reference

## Classes

- class [std::\\_\\_ctype\\_abstract\\_base<\\_CharT>](#)
- class [std::ctype<\\_CharT>](#)
- class [std::ctype<char>](#)
- class [std::ctype<wchar\\_t>](#)
- class [std::ctype\\_byname<\\_CharT>](#)
- class [std::ctype\\_byname<char>](#)
- class [std::num\\_get<\\_CharT, \\_InIter>](#)
- class [std::num\\_put<\\_CharT, \\_OutIter>](#)
- class [std::numpunct<\\_CharT>](#)
- class [std::numpunct\\_byname<\\_CharT>](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_NUM_CXX11_FACETS`
- `#define _GLIBCXX_NUM_FACETS`
- `#define _GLIBCXX_NUM_UNICODE_FACETS`

## Functions

- `template<typename _CharT >`  
`_CharT * std::__add_grouping (_CharT * __s, _CharT __sep, const char * __gbeg, size_t __gsize, const _CharT`  
`* __first, const _CharT * __last)`
- `template<typename _Tp >`  
`void std::__convert_to_v (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<>`  
`void std::__convert_to_v (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()`
- `template<typename _CharT >`  
`ostreambuf_iterator< _CharT > std::__write (ostreambuf_iterator< _CharT > __s, const _CharT * __ws, int`  
`__len)`
- `template<typename _CharT, typename _Outlter >`  
`_Outlter std::__write (_Outlter __s, const _CharT * __ws, int __len)`
- `template<typename _CharT >`  
`bool std::isalnum (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isalpha (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isblank (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::iscntrl (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isdigit (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isgraph (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::islower (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isprint (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale & __loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale & __loc)`

- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`

#### 5.349.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.350 locale\_facets.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define \_LOCALE\_FACETS\_TCC`

### Functions

- `template<typename _CharT >`  
`_CharT * std::\_\_add\_grouping (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)`
- `template<typename _CharT, typename _ValueT >`  
`int std::\_\_int\_to\_char (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)`
- `bool std::\_\_verify\_grouping (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()`

#### 5.350.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.351 `locale_facets_nonio.h` File Reference

#### Classes

- class `std::messages<_CharT>`
- struct `std::messages_base`
- class `std::messages_byname<_CharT>`
- class `std::money_base`
- class `std::money_get<_CharT, _InIter>`
- class `std::money_put<_CharT, _OutIter>`
- class `std::moneypunct<_CharT, _Intl>`
- class `std::moneypunct_byname<_CharT, _Intl>`
- class `std::time_base`
- class `std::time_get<_CharT, _InIter>`
- class `std::time_get_byname<_CharT, _InIter>`
- class `std::time_put<_CharT, _OutIter>`
- class `std::time_put_byname<_CharT, _OutIter>`

#### Namespaces

- `std`

#### 5.351.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.352 `locale_facets_nonio.tcc` File Reference

#### Namespaces

- `std`

#### Macros

- `#define _LOCALE_FACETS_NONIO_TCC`

#### 5.352.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

## 5.353 localefwd.h File Reference

## Classes

- class [std::codecvt<\\_InternT, \\_ExternT, \\_StateT >](#)
- class [std::codecvt\\_byname<\\_InternT, \\_ExternT, \\_StateT >](#)
- class [std::collate<\\_CharT >](#)
- class [std::collate\\_byname<\\_CharT >](#)
- class [std::ctype<\\_CharT >](#)
- class [std::ctype\\_byname<\\_CharT >](#)
- class [std::messages<\\_CharT >](#)
- class [std::messages\\_byname<\\_CharT >](#)
- class [std::money\\_get<\\_CharT, \\_InIter >](#)
- class [std::money\\_put<\\_CharT, \\_OutIter >](#)
- class [std::moneypunct<\\_CharT, \\_Intl >](#)
- class [std::moneypunct\\_byname<\\_CharT, \\_Intl >](#)
- class [std::num\\_get<\\_CharT, \\_InIter >](#)
- class [std::num\\_put<\\_CharT, \\_OutIter >](#)
- class [std::numpunct<\\_CharT >](#)
- class [std::numpunct\\_byname<\\_CharT >](#)
- class [std::time\\_get<\\_CharT, \\_InIter >](#)
- class [std::time\\_get\\_byname<\\_CharT, \\_InIter >](#)
- class [std::time\\_put<\\_CharT, \\_OutIter >](#)
- class [std::time\\_put\\_byname<\\_CharT, \\_OutIter >](#)

## Namespaces

- [std](#)

## Functions

- [template<typename \\_Facet >  
bool std::has\\_facet \(const locale &\\_\\_loc\) throw \(\)](#)
- [template<typename \\_CharT >  
bool std::isalnum \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::isalpha \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::isblank \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::iscntrl \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::isdigit \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::isgraph \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::islower \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)
- [template<typename \\_CharT >  
bool std::isprint \(\\_CharT \\_\\_c, const locale &\\_\\_loc\)](#)

- `template<typename _CharT >`  
`bool std::ispunct (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isspace (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isupper (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`bool std::isxdigit (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::tolower (_CharT __c, const locale &__loc)`
- `template<typename _CharT >`  
`_CharT std::toupper (_CharT __c, const locale &__loc)`
- `template<typename _Facet >`  
`const _Facet & std::use\_facet (const locale &__loc)`

#### 5.353.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.354 losertree.h File Reference

#### Classes

- `struct \_\_gnu\_parallel::\_LoserTreePointerBase< _Tp, _Compare >::_Loser`
- `struct \_\_gnu\_parallel::\_LoserTreeBase< _Tp, _Compare >::_Loser`
- `class \_\_gnu\_parallel::\_LoserTree< __stable, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTree< false, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreeBase< _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointer< __stable, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointer< false, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointerBase< _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointerUnguarded< __stable, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointerUnguarded< false, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreePointerUnguardedBase< _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreeUnguarded< __stable, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreeUnguarded< false, _Tp, _Compare >`
- `class \_\_gnu\_parallel::\_LoserTreeUnguardedBase< _Tp, _Compare >`

#### Namespaces

- `\_\_gnu\_parallel`

#### 5.354.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.



## 5.355 lu\_counter\_metadata.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_metadata< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)
- class [\\_\\_gnu\\_pbds::detail::lu\\_counter\\_policy\\_base< Size\\_Type >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.355.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

## 5.356 lu\_map\_.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::lu\\_map< Key, Mapped, Eq\\_Fn, \\_Alloc, Update\\_Policy >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_GEN_POS`
- `#define PB_DS_LU_NAME`
- `#define PB_DS_LU_TRAITS_BASE`

#### 5.356.1 Detailed Description

Contains a list update map.

## 5.357 macros.h File Reference

### Macros

- `#define __glibcxx_check_bucket_index(_N)`
- `#define __glibcxx_check_can_decrement_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_can_increment(_First, _Size)`
- `#define __glibcxx_check_can_increment_range(_First1, _Last1, _First2)`
- `#define __glibcxx_check_equal_allocs(_This, _Other)`
- `#define __glibcxx_check_erase(_Position)`
- `#define __glibcxx_check_erase_after(_Position)`
- `#define __glibcxx_check_erase_range(_First, _Last)`
- `#define __glibcxx_check_erase_range_after(_First, _Last)`
- `#define __glibcxx_check_heap(_First, _Last)`
- `#define __glibcxx_check_heap_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_insert(_Position)`
- `#define __glibcxx_check_insert_after(_Position)`
- `#define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)`
- `#define __glibcxx_check_irreflexive(_First, _Last)`
- `#define __glibcxx_check_irreflexive2(_First, _Last)`
- `#define __glibcxx_check_irreflexive_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_irreflexive_pred2(_First, _Last, _Pred)`
- `#define __glibcxx_check_max_load_factor(_F)`
- `#define __glibcxx_check_non_empty_range(_First, _Last)`
- `#define __glibcxx_check_nonempty()`
- `#define __glibcxx_check_partitioned_lower(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_partitioned_upper(_First, _Last, _Value)`
- `#define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)`
- `#define __glibcxx_check_self_move_assign(_Other)`
- `#define __glibcxx_check_sorted(_First, _Last)`
- `#define __glibcxx_check_sorted_pred(_First, _Last, _Pred)`
- `#define __glibcxx_check_sorted_set(_First1, _Last1, _First2)`
- `#define __glibcxx_check_sorted_set_pred(_First1, _Last1, _First2, _Pred)`
- `#define __glibcxx_check_string(_String)`
- `#define __glibcxx_check_string_len(_String, _Len)`
- `#define __glibcxx_check_subscript(_N)`
- `#define __glibcxx_check_valid_constructor_range(_First, _Last)`
- `#define __glibcxx_check_valid_range(_First, _Last)`
- `#define __glibcxx_check_valid_range2(_First, _Last, _Dist)`
- `#define __glibcxx_check_valid_range_at(_First, _Last, _File, _Line, _Func)`
- `#define __GLIBCXX_DEBUG_VERIFY(_Cond, _ErrMsg)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT(_Cond, _ErrMsg, _File, _Line)`
- `#define __GLIBCXX_DEBUG_VERIFY_AT_F(_Cond, _ErrMsg, _File, _Line, _Func)`
- `#define __GLIBCXX_DEBUG_VERIFY_COND_AT(_Cond, _ErrMsg, _File, _Line, _Func)`

### 5.357.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.357.2 Macro Definition Documentation

### 5.357.2.1 `__glibcxx_check_erase`

```
#define __glibcxx_check_erase(
 _Position)
```

Verify that we can erase the element referenced by the iterator `_Position`. We can erase the element if the `_Position` iterator is dereferenceable and references this sequence.

Definition at line 216 of file macros.h.

### 5.357.2.2 `__glibcxx_check_erase_after`

```
#define __glibcxx_check_erase_after(
 _Position)
```

Verify that we can erase the element after the iterator `_Position`. We can erase the element if the `_Position` iterator is before a dereferenceable one and references this sequence.

Definition at line 230 of file macros.h.

### 5.357.2.3 `__glibcxx_check_erase_range`

```
#define __glibcxx_check_erase_range(
 _First,
 _Last)
```

Verify that we can erase the elements in the iterator range `[_First, _Last)`. We can erase the elements if `[_First, _Last)` is a valid iterator range within this sequence.

Definition at line 244 of file macros.h.

### 5.357.2.4 `__glibcxx_check_erase_range_after`

```
#define __glibcxx_check_erase_range_after(
 _First,
 _Last)
```

Verify that we can erase the elements in the iterator range `(_First, _Last)`. We can erase the elements if `(_First, _Last)` is a valid iterator range within this sequence.

Definition at line 256 of file macros.h.

**5.357.2.5 \_\_glibcxx\_check\_heap\_pred**

```
#define __glibcxx_check_heap_pred(
 _First,
 _Last,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is a heap w.r.t. the predicate `_Pred`.

Definition at line 441 of file macros.h.

**5.357.2.6 \_\_glibcxx\_check\_insert**

```
#define __glibcxx_check_insert(
 _Position)
```

Verify that we can insert into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 150 of file macros.h.

**5.357.2.7 \_\_glibcxx\_check\_insert\_after**

```
#define __glibcxx_check_insert_after(
 _Position)
```

Verify that we can insert into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the iterator is a `_Safe_iterator`.

Definition at line 167 of file macros.h.

**5.357.2.8 \_\_glibcxx\_check\_insert\_range**

```
#define __glibcxx_check_insert_range(
 _Position,
 _First,
 _Last,
 _Dist)
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` with the iterator `_Position`. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 184 of file macros.h.

### 5.357.2.9 \_\_glibcxx\_check\_insert\_range\_after

```
#define __glibcxx_check_insert_range_after(
 _Position,
 _First,
 _Last,
 _Dist)
```

Verify that we can insert the values in the iterator range `[_First, _Last)` into `*this` after the iterator `_Position`. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range `[_First, _Last)` is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a `_Safe_sequence` and the `_Position` iterator is a `_Safe_iterator`.

Definition at line 203 of file `macros.h`.

### 5.357.2.10 \_\_glibcxx\_check\_partitioned\_lower

```
#define __glibcxx_check_partitioned_lower(
 _First,
 _Last,
 _Value)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value`.

Definition at line 385 of file `macros.h`.

### 5.357.2.11 \_\_glibcxx\_check\_partitioned\_lower\_pred

```
#define __glibcxx_check_partitioned_lower_pred(
 _First,
 _Last,
 _Value,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 407 of file `macros.h`.

### 5.357.2.12 \_\_glibcxx\_check\_partitioned\_upper\_pred

```
#define __glibcxx_check_partitioned_upper_pred(
 _First,
 _Last,
 _Value,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is partitioned w.r.t. the value `_Value` and predicate `_Pred`.

Definition at line 420 of file `macros.h`.

### 5.357.2.13 `__glibcxx_check_sorted_pred`

```
#define __glibcxx_check_sorted_pred(
 _First,
 _Last,
 _Pred)
```

Verify that the iterator range `[_First, _Last)` is sorted by the predicate `_Pred`.

Definition at line 350 of file `macros.h`.

### 5.357.2.14 `_GLIBCXX_DEBUG_VERIFY_COND_AT`

```
#define _GLIBCXX_DEBUG_VERIFY_COND_AT(
 _Cond,
 _ErrMsg,
 _File,
 _Line,
 _Func)
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

Definition at line 50 of file `macros.h`.

## 5.358 `malloc_allocator.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::malloc\\_allocator<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.358.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.359 `map` File Reference

### Macros

- `#define _GLIBCXX_MAP`

## 5.359.1 Detailed Description

This is a Standard C++ Library header.

## 5.360 map File Reference

## Classes

- class [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)
- class [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_MAP`

## 5.360.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.361 map File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_MAP`

## Typedefs

- `template<typename _Key , typename _Tp , typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::map = std::map< _Key, _Tp, _Compare, polymorphic_allocator< pair< const _Key, _Tp > >>`
- `template<typename _Key , typename _Tp , typename _Compare = less<_Key>>  
using std::experimental::fundamentals_v2::pmr::multimap = std::multimap< _Key, _Tp, _Compare, polymorphic_allocator< pair< const _Key, _Tp > >>`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (map< _Key, _Tp, _Compare, _Alloc > &__cont, _↔`  
`Predicate __pred)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _↔`  
`Predicate __pred)`

### 5.361.1 Detailed Description

This is a TS C++ Library header.

## 5.362 map.h File Reference

### Classes

- class [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >](#)

### Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key,`  
`_Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare,`  
`_Allocator > &__rhs) noexcept(/*conditional */)`



## 5.362.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.363 mask\_array.h File Reference

## Classes

- class [std::mask\\_array<\\_Tp>](#)

## Namespaces

- [std](#)

## Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

## 5.363.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.364 mask\_based\_range\_hashing.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::mask\\_based\\_range\\_hashing<Size\\_Type>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.364.1 Detailed Description

Contains a range hashing policy base.

## 5.365 math.h File Reference

## 5.365.1 Detailed Description

This is a Standard C++ Library header.

## 5.366 memory File Reference

### Namespaces

- [std](#)

### Macros

- `#define _GLIBCXX_MEMORY`

### Enumerations

- enum [std::pointer\\_safety](#) { **relaxed**, **preferred**, **strict** }

### Functions

- void \* [std::align](#) (size\_t \_\_align, size\_t \_\_size, void \*&\_\_ptr, size\_t &\_\_space) noexcept
- void [std::declare\\_no\\_pointers](#) (char \*, size\_t)
- void [std::declare\\_reachable](#) (void \*)
- pointer\_safety [std::get\\_pointer\\_safety](#) () noexcept
- void [std::undeclare\\_no\\_pointers](#) (char \*, size\_t)
- template<typename \_Tp >  
\_Tp \* [std::undeclare\\_reachable](#) (\_Tp \* \_\_p)

#### 5.366.1 Detailed Description

This is a Standard C++ Library header.

## 5.367 memory File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::temporary\\_buffer](#)< [\\_ForwardIterator](#), [\\_Tp](#) >

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _EXT_MEMORY`

## Functions

- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _InputIter __first, _Size __count, ↵`  
`_ForwardIter __result, std::input_iterator_tag)`
- `template<typename _RandomAccessIter, typename _Size, typename _ForwardIter >`  
`std::pair< _RandomAccessIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _RandomAccessIter ↵`  
`_first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n ( _InputIter __first, _Size __count, ↵`  
`_ForwardIter __result)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator >`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count,`  
`_ForwardIter __result, _Allocator __alloc)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp >`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::__uninitialized_copy_n_a ( _InputIter __first, _Size __count,`  
`_ForwardIter __result, std::allocator< _Tp >)`
- `template<typename _InputIter, typename _Size, typename _ForwardIter >`  
`std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n ( _InputIter __first, _Size __count, ↵`  
`_ForwardIter __result)`

## 5.367.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 5.368 memory File Reference

## Namespaces

- `std`
- `std::experimental`

## Macros

- `#define __cpp_lib_experimental_observer_ptr`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY`

## Functions

- `template<typename _Tp >`  
`observer_ptr< _Tp > std::experimental::fundamentals_v2::make_observer ( _Tp *__p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator!= (observer_ptr< _Tp > __p1, observer_ptr< _Up > ↵`  
`__p2)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (observer_ptr< _Tp > __p, nullptr_t) noexcept`

- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator== (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (observer_ptr< _Tp > __p, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (nullptr_t, observer_ptr< _Tp > __p) noexcept`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp, typename _Up >`  
`bool std::experimental::fundamentals_v2::operator>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept`

#### 5.368.1 Detailed Description

This is a TS C++ Library header.

### 5.369 memory\_resource File Reference

#### Macros

- `#define _GLIBCXX_MEMORY_RESOURCE`

#### 5.369.1 Detailed Description

This is a Standard C++ Library header.

### 5.370 memory\_resource File Reference

#### Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_memory_resources`
- `#define _GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE`

## Typedefs

- `template<typename _Alloc >`  
using **`std::experimental::fundamentals_v2::pmr::resource_adaptor`** = `__resource_adaptor_imp< typename allocator_traits< _Alloc >::template rebind_alloc< char > >`

## Functions

- `memory_resource * std::experimental::fundamentals\_v2::pmr::get\_default\_resource ()` noexcept
- `memory_resource * std::experimental::fundamentals_v2::pmr::new_delete_resource ()` noexcept
- `memory_resource * std::experimental::fundamentals_v2::pmr::null_memory_resource ()` noexcept
- `bool std::experimental::fundamentals_v2::pmr::operator!= (const memory_resource &__a, const memory_resource &__b)` noexcept
- `template<class _Tp1, class _Tp2 >`  
`bool std::experimental::fundamentals_v2::pmr::operator!= (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b)` noexcept
- `bool std::experimental::fundamentals_v2::pmr::operator== (const memory_resource &__a, const memory_resource &__b)` noexcept
- `template<class _Tp1, class _Tp2 >`  
`bool std::experimental::fundamentals_v2::pmr::operator== (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b)` noexcept
- `memory_resource * std::experimental::fundamentals\_v2::pmr::set\_default\_resource (memory_resource *__r)` noexcept

### 5.370.1 Detailed Description

This is a TS C++ Library header.

### 5.370.2 Function Documentation

#### 5.370.2.1 `get_default_resource()`

```
memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () [inline],
[noexcept]
```

Get the current default resource.

Definition at line 549 of file experimental/memory\_resource.

### 5.370.2.2 `set_default_resource()`

```
memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (
 memory_resource * __r) [inline], [noexcept]
```

Change the default resource and return the previous one.

Definition at line 554 of file `experimental/memory_resource`.

## 5.371 `memoryfwd.h` File Reference

### Classes

- class [std::allocator<\\_Tp>](#)
- struct [std::uses\\_allocator<\\_Tp, \\_Alloc>](#)

### Namespaces

- [std](#)

### 5.371.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.372 `merge.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2,`  
`_RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↵`  
`__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>`  
`_OutputIterator \_\_gnu\_parallel::\_\_merge\_advance\_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↵`  
`__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↵`  
`begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator\_traits<_RAIter1>::difference_type __↵`  
`max_length, _Compare __comp)`
- `template<typename _RAIter1, typename _RAIter3, typename _Compare>`  
`_RAIter3 \_\_gnu\_parallel::\_\_parallel\_merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &↵`  
`begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator\_traits<_RAIter1>::difference_type __↵`  
`max_length, _Compare __comp)`

## 5.372.1 Detailed Description

Parallel implementation of `std::merge()`. This file is a GNU parallel extension to the Standard C++ Library.

5.373 `messages_members.h` File Reference

## Namespaces

- [std](#)

## 5.373.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

5.374 `mod_based_range_hashing.hpp` File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::mod\\_based\\_range\\_hashing< Size\\_Type >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.374.1 Detailed Description

Contains a range hashing policy base.

5.375 `move.h` File Reference

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_FORWARD(__Tp, __val)`
- `#define _GLIBCXX_MOVE(__val)`

## Functions

- `template<typename _Tp >`  
`constexpr _Tp * std::\_\_addressof (_Tp &__r) noexcept`
- `template<typename _Tp, typename _Up = _Tp>`  
`constexpr _Tp std::__exchange (_Tp &__obj, _Up &&__new_val)`
- `template<typename _Tp >`  
`constexpr _Tp * std::addressof (_Tp &__r) noexcept`
- `template<typename _Tp >`  
`const _Tp * std::addressof (const _Tp &&)=delete`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &__t) noexcept`
- `template<typename _Tp >`  
`constexpr _Tp && std::forward (typename std::remove\_reference< _Tp >::type &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr std::remove\_reference< _Tp >::type && std::move (_Tp &&__t) noexcept`
- `template<typename _Tp >`  
`constexpr conditional< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && >::type std::move\_if\_noexcept`  
`( _Tp &__x) noexcept`
- `template<typename _Tp >`  
`constexpr enable_if< __and< __not< __is_tuple_like< _Tp >, is_move_constructible< _Tp >, is_move_↔`  
`_assignable< _Tp > >::value >::type std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_↔`  
`_move_assignable< _Tp > >`
- `template<typename _Tp, size_t _Nm>`  
`constexpr enable_if< __is_swappable< _Tp >::value >::type std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm])`  
`noexcept(/*conditional */)`

### 5.375.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

### 5.376 `mt_allocator.h` File Reference

#### Classes

- `struct \_\_gnu\_cxx::\_\_common\_pool\_policy< _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc< _Tp, _Poolp >`
- `class \_\_gnu\_cxx::\_\_mt\_alloc\_base< _Tp >`
- `struct \_\_gnu\_cxx::\_\_per\_type\_pool\_policy< _Tp, _PoolTp, _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< _Thread >`
- `class \_\_gnu\_cxx::\_\_pool< false >`
- `class \_\_gnu\_cxx::\_\_pool< true >`
- `struct \_\_gnu\_cxx::\_\_pool\_base`

#### Namespaces

- `\_\_gnu\_cxx`



## Macros

- `#define __thread_default`

## Typedefs

- `typedef void(* __gnu_cxx::__destroy_handler) (void *)`

## Functions

- `template<typename _Tp, typename _Poolp >  
bool __gnu_cxx::operator!= (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`
- `template<typename _Tp, typename _Poolp >  
bool __gnu_cxx::operator== (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)`

## 5.376.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.377 multimap.h File Reference

## Classes

- `class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`

## Namespaces

- `std`
- `std::__debug`

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator!= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator< (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator<= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator== (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator> (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
bool std::__debug::operator>= (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Allocator >  
void std::__debug::swap (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)`

### 5.377.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.378 multiseq\_selection.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_Lexicographic<\\_T1, \\_T2, \\_Compare>](#)
- class [\\_\\_gnu\\_parallel::\\_LexicographicReverse<\\_T1, \\_T2, \\_Compare>](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define __S(__i)`
- `#define __S(__i)`

### Functions

- `template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>`  
`void \_\_gnu\_parallel::multiseq\_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator\_traits< typename`  
`std::iterator\_traits< _RanSeqs >::value_type::first_type >::value_type >())`
- `template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>`  
`_Tp \_\_gnu\_parallel::multiseq\_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank,`  
`_RankType &__offset, _Compare __comp=std::less< _Tp >())`

### 5.378.1 Detailed Description

Functions to find elements of a certain global \_\_rank in multiple sorted sequences. Also serves for splitting such sequence sets.

The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171-177, 1991.

This file is a GNU parallel extension to the Standard C++ Library.

## 5.379 multiset.h File Reference

## Classes

- class [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >](#)

## Namespaces

- [std](#)
- [std::\\_\\_debug](#)

## Functions

- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator!= \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator< \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator<= \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator== \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator> \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > bool std::\\_\\_debug::operator>= \(const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_lhs, const multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_rhs\)](#)
- [template<typename \\_Key, typename \\_Compare, typename \\_Allocator > void std::\\_\\_debug::swap \(multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_x, multiset< \\_Key, \\_Compare, \\_Allocator > &\\_\\_y\) noexcept\(\*/\\*conditional \\*/\*\)](#)

## 5.379.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.380 multiway\_merge.h File Reference

## Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_\\_stable, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAlterIterator, \\_RAlter3, \\_DifferenceTp, \\_Compare >](#)
- class [\\_\\_gnu\\_parallel::GuardedIterator< \\_RAlter, \\_Compare >](#)
- struct [\\_\\_gnu\\_parallel::LoserTreeTraits< \\_Tp >](#)
- struct [\\_\\_gnu\\_parallel::SamplingSorter< \\_\\_stable, \\_RAlter, \\_StrictWeakOrdering >](#)
- struct [\\_\\_gnu\\_parallel::SamplingSorter< false, \\_RAlter, \\_StrictWeakOrdering >](#)

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_PARALLEL\_DECISION(__a, __b, __c, __d)`
- `#define \_GLIBCXX\_PARALLEL\_LENGTH(__s)`
- `#define \_GLIBCXX\_PARALLEL\_MERGE\_3\_CASE(__a, __b, __c, __c0, __c1)`
- `#define \_GLIBCXX\_PARALLEL\_MERGE\_4\_CASE(__a, __b, __c, __d, __c0, __c1, __c2)`

## Functions

- `template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare >  
_OutputIterator \_\_gnu\_parallel::merge\_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
<_RAIter3 \_\_gnu\_parallel::sequential\_multiway\_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator\_traits< typename std::iterator\_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, \_\_gnu\_parallel::sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut \_\_gnu\_parallel::multiway\_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_3\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_4\_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >  
void \_\_gnu\_parallel::multiway\_merge\_exact\_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >  
_RAIter3 \_\_gnu\_parallel::multiway\_merge\_loser\_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)`

- `template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator <-`  
`__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIter<-`  
`Iterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare >`  
`_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator <-`  
`__seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIter<-`  
`Iterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)`
- `template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType >`  
`void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator <-`  
`__seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector<`  
`std::pair< _DifferenceType, _DifferenceType > > *__pieces)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel<-`  
`tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter,`  
`typename _Compare >`  
`_RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end,`  
`_RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num<-`  
`threads)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel<-`  
`tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator <-`  
`__seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >`  
`_RAIterOut __gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIter<-`  
`PairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)`

- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, gnu\_parallel::exact\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling\_tag __tag)`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel\_tag __tag=parallel_tag(0))`
- `template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare >  
_RAIterOut gnu_parallel::stable_multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default\_parallel\_tag __tag)`

### 5.380.1 Detailed Description

Implementation of sequential and parallel multiway merge.

Explanations on the high-speed merging routines in the appendix of

P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.

This file is a GNU parallel extension to the Standard C++ Library.

### 5.380.2 Macro Definition Documentation

#### 5.380.2.1 \_GLIBCXX\_PARALLEL\_LENGTH

```
#define _GLIBCXX_PARALLEL_LENGTH(
 __s)
```

Length of a sequence described by a pair of iterators.

Definition at line 54 of file multiway\_merge.h.

## 5.381 multiway\_mergesort.h File Reference

### Classes

- `struct gnu\_parallel::Piece< _DifferenceTp >`
- `struct gnu\_parallel::PMWMSortingData< _RAIter >`
- `struct gnu\_parallel::SplitConsistently< exact, _RAIter, _Compare, _SortingPlacesIterator >`
- `struct gnu\_parallel::SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >`
- `struct gnu\_parallel::SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >`

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >`  
`void \_\_gnu\_parallel::\_\_determine\_samples (_PMWMSortingData< _RAIter > *__sd, _DifferenceTp __num←  
 _samples)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex  
 __num_threads)`
- `template<bool __stable, bool __exact, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::parallel\_sort\_mwms\_pu (_PMWMSortingData< _RAIter > *__sd, _Compare &__comp)`

## 5.381.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

## 5.382 mutex File Reference

## Classes

- struct [std::once\\_flag](#)
- class [std::recursive\\_mutex](#)
- class [std::recursive\\_timed\\_mutex](#)
- class [std::timed\\_mutex](#)

## Namespaces

- [std](#)

## Macros

- `#define \_GLIBCXX\_MUTEX`

## Functions

- `template<typename _Callable, typename... _Args>`  
`void std::call\_once (once_flag &__once, _Callable &&__f, _Args &&... __args)`
- `template<typename _L1, typename _L2, typename... _L3>`  
`void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)`
- `template<typename _Lock1, typename _Lock2, typename... _Lock3>`  
`int std::try\_lock (_Lock1 &__l1, _Lock2 &__l2, _Lock3 &... __l3)`

### 5.382.1 Detailed Description

This is a Standard C++ Library header.

## 5.383 `nested_exception.h` File Reference

### Classes

- class [std::nested\\_exception](#)

### Namespaces

- [std](#)

### Functions

- `template<typename _Ex >`  
`void std::rethrow\_if\_nested (const _Ex &__ex)`
- `template<typename _Tp >`  
`void std::throw\_with\_nested (_Tp &&__t)`

### 5.383.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<exception>`.

## 5.384 `net.h` File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### 5.384.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/net>`.

## 5.385 `new` File Reference

### Classes

- class [std::bad\\_alloc](#)



## Namespaces

- [std](#)

## Typedefs

- typedef void(\* [std::new\\_handler](#)) ()

## Functions

- new\_handler [std::get\\_new\\_handler](#) () noexcept
- new\_handler [std::set\\_new\\_handler](#) (new\_handler) throw ()
  
- void \* [operator new](#) (std::size\_t)
- void \* [operator new\[\]](#) (std::size\_t)
- void [operator delete](#) (void \*) noexcept
- void [operator delete\[\]](#) (void \*) noexcept
- void \* [operator new](#) (std::size\_t, const std::nothrow\_t &) noexcept
- void \* [operator new\[\]](#) (std::size\_t, const std::nothrow\_t &) noexcept
- void [operator delete](#) (void \*, const std::nothrow\_t &) noexcept
- void [operator delete\[\]](#) (void \*, const std::nothrow\_t &) noexcept
- void \* [operator new](#) (std::size\_t, void \*\_\_p) noexcept
- void \* [operator new\[\]](#) (std::size\_t, void \*\_\_p) noexcept
- void [operator delete](#) (void \*, void \*) noexcept
- void [operator delete\[\]](#) (void \*, void \*) noexcept

## Variables

- const nothrow\_t **std::nothrow**

### 5.385.1 Detailed Description

This is a Standard C++ Library header.

The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see [https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic\\_memory.html](https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic_memory.html) for more.

### 5.385.2 Function Documentation

**5.385.2.1 operator delete()** [1/3]

```
void operator delete (
 void *) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.2 operator delete()** [2/3]

```
void operator delete (
 void * ,
 const std::nothrow_t &) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.3 operator delete()** [3/3]

```
void operator delete (
 void * ,
 void *) [inline], [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 180 of file `new`.

**5.385.2.4 operator delete[]()** [1/3]

```
void operator delete[] (
 void *) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.5 operator delete[]()** [2/3]

```
void operator delete[] (
 void * ,
 const std::nothrow_t &) [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.6 operator delete[]()** [3/3]

```
void operator delete[] (
 void * ,
 void *) [inline], [noexcept]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 181 of file new.

**5.385.2.7 operator new()** [1/3]

```
void* operator new (
 std::size_t)
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.8 operator new()** [2/3]

```
void* operator new (
 std::size_t ,
 const std::nothrow_t &) [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.9 operator new()** [3/3]

```
void* operator new (
 std::size_t ,
 void * __p) [inline], [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 174 of file new.

**5.385.2.10 operator new[]()** [1/3]

```
void* operator new[] (
 std::size_t)
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.11 operator new[]()** [2/3]

```
void* operator new[] (
 std::size_t ,
 const std::nothrow_t &) [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

**5.385.2.12 operator new[]()** [3/3]

```
void* operator new[] (
 std::size_t ,
 void * __p) [inline], [nothrow]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

Definition at line 176 of file `new`.

## 5.386 new\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::new\\_allocator< \\_Tp >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

#### 5.386.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.387 node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_< \\_Value, \\_Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.387.1 Detailed Description

Contains an implementation struct for this type of heap's node.

## 5.388 node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node\\_< Value\\_Type, Metadata, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.388.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.389 node.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node\\_](#)< Value\_Type, Metadata, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.389.1 Detailed Description

Contains an implementation struct for splay\_tree\_'s node.

## 5.390 node\_handle.h File Reference

#### 5.390.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map,set,unordered_map,unordered_set>`.

## 5.391 node\_iterators.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, \_Alloc >
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it\\_](#)< Node, Const\_Iterator, Iterator, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC`
- `#define PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC`

#### 5.391.1 Detailed Description

Contains an implementation class for bin\_search\_tree\_.

## 5.392 node\_iterators.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_const\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::ov\\_tree\\_node\\_it\\_< Value\\_Type, Metadata\\_Type, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC`
- `#define PB_DS_OV_TREE_NODE_ITERATOR_C_DEC`

#### 5.392.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 5.393 node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.393.1 Detailed Description

Contains an implementation class for trees.

## 5.394 node\_metadata\_selector.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, \\_BTp >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, false >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_metadata\\_helper< Node\\_Update, true >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >](#)



## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.394.1 Detailed Description

Contains an implementation class for tries.

## 5.395 null\_node\_metadata.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::dumnode\\_const\\_iterator](#)< Key, Data, \_Alloc >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.395.1 Detailed Description

Contains an implementation class for tree-like classes.

## 5.396 numbers File Reference

## Macros

- `#define \_GLIBCXX\_NUMBERS`

## 5.396.1 Detailed Description

This is a Standard C++ Library header.

## 5.397 numeric File Reference

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Macros

- `#define _GLIBCXX_NUMERIC`

## Functions

- `template<typename _Up, typename _Tp >`  
`constexpr _Up std::__detail::__absu (_Tp __val)`
- `template<typename _Up >`  
`void std::__detail::__absu (bool)=delete`
- `template<typename _Tp >`  
`constexpr _Tp std::__detail::__gcd (_Tp __m, _Tp __n)`
- `template<typename _Tp >`  
`constexpr _Tp std::__detail::__lcm (_Tp __m, _Tp __n)`

### 5.397.1 Detailed Description

This is a Standard C++ Library header.

## 5.398 numeric File Reference

### Namespaces

- [`\_\_gnu\_cxx`](#)

## Macros

- `#define _EXT_NUMERIC`

## Functions

- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`
- `template<typename _Tp, typename _Integer, typename _MonoidOperation >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)`
- `template<typename _Tp, typename _Integer >`  
`_Tp \_\_gnu\_cxx::power (_Tp __x, _Integer __n)`

### 5.398.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 5.399 numeric File Reference

## Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

## Macros

- `#define \_GLIBCXX\_PARALLEL\_NUMERIC\_H`

## Functions

- `template<typename _Iter, typename _Tp, typename _IteratorTag >  
_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter __begin, _Iter __end, _Tp __init, _IteratorTag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation, typename _IteratorTag >  
_Tp std::\_\_parallel::\_\_accumulate\_switch (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOperation >  
_Tp std::\_\_parallel::\_\_accumulate\_switch (_RAIter __begin, _RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >  
_OutputIterator std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator std::\_\_parallel::\_\_adjacent\_difference\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag)`
- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >  
_Tp std::\_\_parallel::\_\_inner\_product\_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::\_\_Parallelism __parallelism_tag, \_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2 >  
_Tp std::\_\_parallel::\_\_inner\_product\_switch (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2 >  
_OutputIterator std::\_\_parallel::\_\_partial\_sum\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >  
_OutputIterator std::\_\_parallel::\_\_partial\_sum\_switch (_Iter __begin, _Iter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >  
_Tp std::\_\_parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >  
_Tp std::\_\_parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >  
_Tp std::\_\_parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::\_\_Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >  
_Tp std::\_\_parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOperation >`  
`_Tp std::__parallel::accumulate (_Iter __begin, _Iter __end, _Tp __init, _BinaryOperation __binary_op)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _↵`  
`BinaryOperation __bin_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _↵`  
`BinaryOperation __binary_op, __gnu_parallel::__Parallelism __parallelism_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::adjacent_difference (_Iter __begin, _Iter __end, _OutputIterator __result, _↵`  
`BinaryOperation __binary_op)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, __gnu_parallel::__Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2,`  
`__gnu_parallel::__Parallelism)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result,`  
`__gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵`  
`Operation __bin_op, __gnu_parallel::__sequential_tag)`
- `template<typename _Iter, typename _OutputIterator >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator std::__parallel::partial_sum (_Iter __begin, _Iter __end, _OutputIterator __result, _Binary↵`  
`Operation __binary_op)`

### 5.399.1 Detailed Description

Parallel STL function calls corresponding to `stl_numeric.h`. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

## 5.400 numeric File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_gcd_lcm`
- `#define _GLIBCXX_EXPERIMENTAL_NUMERIC`

### Functions

- `template<typename _Mn, typename _Nn >`  
`constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals\_v2::gcd (_Mn __m, _Nn __n) noexcept`
- `template<typename _Mn, typename _Nn >`  
`constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals\_v2::lcm (_Mn __m, _Nn __n)`

#### 5.400.1 Detailed Description

This is a TS C++ Library header.

#### 5.400.2 Function Documentation

##### 5.400.2.1 gcd()

```
template<typename _Mn, typename _Nn >
constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals_v2::gcd (
 _Mn __m,
 _Nn __n) [noexcept]
```

Greatest common divisor.

Definition at line 57 of file experimental/numeric.

##### 5.400.2.2 lcm()

```
template<typename _Mn, typename _Nn >
constexpr common_type_t<_Mn, _Nn> std::experimental::fundamentals_v2::lcm (
 _Mn __m,
 _Nn __n)
```

Least common multiple.

Definition at line 75 of file experimental/numeric.

## 5.401 `numeric_traits.h` File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define __glibcxx_digits10(_Tp)`
- `#define __glibcxx_floating(_Tp, _Fval, _Dval, _LDval)`
- `#define __glibcxx_max_digits10(_Tp)`
- `#define __glibcxx_max_exponent10(_Tp)`
- `#define __GLIBCXX_INT_N_TRAITS(T, WIDTH)`

### Typedefs

- `template<typename _Tp >`  
`using \_\_gnu\_cxx::\_\_int\_traits = __numeric_traits_integer< _Tp >`

### 5.401.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.402 `numeric_fwd.h` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_parallel](#)

### Functions

- `template<typename _Iter, typename _Tp, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _Tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper, typename _Tag >`  
`_Tp std::__parallel::__accumulate_switch (_Iter, _Iter, _Tp, _BinaryOper, _Tag)`
- `template<typename _RAIter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::__accumulate_switch (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >`  
`_OIter std::__parallel::__adjacent_difference_switch (_Iter, _Iter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism __parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`

- `template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::__inner_product_switch (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, \_\_gnu\_parallel::Parallelism=\_\_gnu\_parallel::parallel\_unbalanced)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2 >`  
`_Tp std::__parallel::__inner_product_switch (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper, typename _Tag1, typename _Tag2 >`  
`_Olter std::__parallel::__partial_sum_switch (_Iter, _Iter, _Olter, _BinaryOper, _Tag1, _Tag2)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper >`  
`_Olter std::__parallel::__partial_sum_switch (_Iter, _Iter, _Olter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Tp, typename _BinaryOper >`  
`_Tp std::__parallel::accumulate (_Iter, _Iter, _Tp, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Olter >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper)`
- `template<typename _Iter, typename _Olter >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _Olter >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Olter, typename _BinaryOper >`  
`_Olter std::__parallel::adjacent_difference (_Iter, _Iter, _Olter, _BinaryOper, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter1, typename _Iter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2 >`  
`_Tp std::__parallel::inner_product (_Iter1, _Iter1, _Iter2, _Tp, _BinaryFunction1, _BinaryFunction2, \_\_gnu\_parallel::Parallelism)`
- `template<typename _Iter, typename _Olter >`  
`_Olter std::__parallel::partial_sum (_Iter, _Iter, _Olter, \_\_gnu\_parallel::sequential\_tag)`

- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper, \_\_gnu\_parallel::sequential\_tag)`
- `template<typename _Iter, typename _OIter >  
_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter __result)`
- `template<typename _Iter, typename _OIter, typename _BinaryOper >  
_OIter std::__parallel::partial_sum (_Iter, _Iter, _OIter, _BinaryOper)`

#### 5.402.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

### 5.403 `omp_loop.h` File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop (_RAIter __begin, _RAIter __end, _Op <←  
__o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter ><←  
::difference_type __bound)`

#### 5.403.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

### 5.404 `omp_loop_static.h` File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_omp\_loop\_static (_RAIter __begin, _RAIter __end,  
_Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter ><←  
::difference_type __bound)`



#### 5.404.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

### 5.405 `opt_random.h` File Reference

#### Namespaces

- [std](#)

#### 5.405.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

### 5.406 `optional` File Reference

#### Macros

- `#define _GLIBCXX_OPTIONAL`

#### 5.406.1 Detailed Description

This is a Standard C++ Library header.

### 5.407 `optional` File Reference

#### Classes

- class [std::experimental::fundamentals\\_v1::bad\\_optional\\_access](#)
- struct [std::hash< experimental::optional< \\_Tp > >](#)
- struct [std::experimental::fundamentals\\_v1::in\\_place\\_t](#)
- struct [std::experimental::fundamentals\\_v1::nullopt\\_t](#)
- class [std::experimental::fundamentals\\_v1::optional< \\_Tp >](#)
- class [std::experimental::fundamentals\\_v1::optional< \\_Tp >](#)

#### Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define __cpp_lib_experimental_optional`
- `#define _GLIBCXX_EXPERIMENTAL_OPTIONAL`

## Variables

- `constexpr in_place_t` [std::experimental::fundamentals\\_v1::in\\_place](#)
- `constexpr nullopt_t` [std::experimental::fundamentals\\_v1::nullopt](#)

### 5.407.1 Detailed Description

This is a TS C++ Library header.

## 5.408 `order_statistics_imp.hpp` File Reference

### 5.408.1 Detailed Description

Contains forward declarations for `order_statistics_key`

## 5.409 `order_statistics_imp.hpp` File Reference

### 5.409.1 Detailed Description

Contains forward declarations for `order_statistics_key`

## 5.410 `os_defines.h` File Reference

## Macros

- `#define __NO_CTYPE`
- `#define _GLIBCXX_NATIVE_THREAD_ID`
- `#define _GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC`

### 5.410.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

## 5.411 ostream File Reference

## Classes

- class [std::basic\\_ostream<\\_CharT, \\_Traits>](#)
- class [std::basic\\_ostream<\\_CharT, \\_Traits>::sentry](#)

## Namespaces

- [std](#)

## Macros

- `#define \_GLIBCXX\_OSTREAM`

## Typedefs

- `template<typename _Tp>`  
`using std::\_\_do\_is\_convertible\_to\_basic\_ostream\_impl = decltype(__is_convertible_to_basic_ostream_↵`  
`test(declval<typename remove_reference<_Tp>::type*>()))`
- `template<typename _Ostream>`  
`using std::\_\_rvalue\_ostream\_type = typename __is_convertible_to_basic_ostream<_Ostream>::__↵`  
`ostream_type`

## Functions

- `template<typename _Ch, typename _Up>`  
`basic_ostream<_Ch, _Up> & std::\_\_is\_convertible\_to\_basic\_ostream\_test (basic_ostream<_Ch, _Up>`  
`*)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::endl (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::ends (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::flush (basic_ostream<_CharT, _Traits> &__os)`
- `template<typename _Ostream, typename _Tp>`  
`enable_if<__and<__not<is_lvalue_reference<_Ostream>>, __is_convertible_to_basic_ostream<↵`  
`_Ostream>, __is_insertable<__rvalue_ostream_type<_Ostream>, const _Tp &>>::value, __rvalue_↵`  
`ostream_type<_Ostream>>::type std::operator<< (_Ostream &&__os, const _Tp &__x)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__out, _CharT __c)`
- `template<typename _CharT, typename _Traits>`  
`basic_ostream<_CharT, _Traits> & std::operator<< (basic_ostream<_CharT, _Traits> &__out, char __c)`

- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)`
- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)`
- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)`
  
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`
- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)`
- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)`
- `template<typename _Traits >`  
`basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)`

#### 5.411.1 Detailed Description

This is a Standard C++ Library header.

### 5.412 ostream.tcc File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _OSTREAM_TCC`

#### Functions

- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)`

## 5.412.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 5.413 ostream\_insert.h File Reference

## Namespaces

- [std](#)

## Functions

- `template<typename _CharT, typename _Traits >`  
`void std::__ostream_fill (basic_ostream< _CharT, _Traits > &__out, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::__ostream_insert (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`
- `template<typename _CharT, typename _Traits >`  
`void std::__ostream_write (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s, streamsize __n)`

## 5.413.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ostream>`.

## 5.414 ov\_tree\_map.hpp File Reference

## Classes

- [class `\_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >`](#)
- [class `\_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >`](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_NODE_ITERATOR_NAME`
- `#define PB_DS_OV_TREE_NAME`
- `#define PB_DS_OV_TREE_TRAITS_BASE`

#### 5.414.1 Detailed Description

Contains an implementation class for `ov_tree`.

### 5.415 `pairing_heap.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::pairing\\_heap](#)< `Value_Type`, `Cmp_Fn`, `_Alloc` >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_P_HEAP_BASE`

#### 5.415.1 Detailed Description

Contains an implementation class for a pairing heap.

### 5.416 `par_loop.h` File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op \_\_gnu\_parallel::\_\_for\_each\_template\_random\_access\_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu  
&__f, _Red __r, _Result __base, _Result &__output, typename std::iterator\_traits< _RAIter >::difference_type  
__bound)`

#### 5.416.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

## 5.417 parallel.h File Reference

### 5.417.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

## 5.418 parse\_numbers.h File Reference

### Namespaces

- [std](#)

### Typedefs

- `template<unsigned long long _Val>`  
using **std::\_\_parse\_int::\_\_ull\_constant** = integral\_constant< unsigned long long, \_Val >
- `template<char... _Digs>`  
using **std::\_\_select\_int::Select\_int** = typename \_Select\_int\_base< \_\_parse\_int::Parse\_int< \_Digs... >::value, unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long >::type

### 5.418.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<chrono>`.

## 5.419 partial\_sum.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum (_Iter __begin, _Iter __end, _OutputIterator __result, \_\_  
_BinaryOperation __bin_op)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_basecase (_Iter __begin, _Iter __end, _OutputIterator`  
`__result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::value_type __value)`
- `template<typename _Iter, typename _OutputIterator, typename _BinaryOperation >`  
`_OutputIterator \_\_gnu\_parallel::\_\_parallel\_partial\_sum\_linear (_Iter __begin, _Iter __end, _OutputIterator \_\_  
__result, _BinaryOperation __bin_op, typename std::iterator\_traits< _Iter >::difference_type __n)`

#### 5.419.1 Detailed Description

Parallel implementation of `std::partial_sum()`, i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

### 5.420 partition.h File Reference

#### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### Macros

- `#define _GLIBCXX_VOLATILE`

#### Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_nth\_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_partial\_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)`
- `template<typename _RAIter, typename _Predicate >`  
`std::iterator\_traits< _RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)`

#### 5.420.1 Detailed Description

Parallel implementation of `std::partition()`, `std::nth_element()`, and `std::partial_sort()`. This file is a GNU parallel extension to the Standard C++ Library.

#### 5.420.2 Macro Definition Documentation

##### 5.420.2.1 \_GLIBCXX\_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.

Definition at line 43 of file `partition.h`.



## 5.421 pat\_trie\_.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_NODE_VALID(X)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_PAT_TRIE_NAME`
- `#define PB_DS_PAT_TRIE_TRAITS_BASE`
- `#define PB_DS_RECURSIVE_COUNT_LEAFS(X)`

## 5.421.1 Detailed Description

Contains an implementation class for a patricia tree.

## 5.422 pat\_trie\_base.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Clter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Head< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)
- class [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::const\\_iterator](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Inode< \\_ATraits, Metadata >::iterator](#)
- struct [\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CONST_IT_C_DEC`
- `#define PB_DS_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_IT_C_DEC`
- `#define PB_DS_ODIR_IT_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC`
- `#define PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC`

### 5.422.1 Detailed Description

Contains the base class for a patricia tree.

## 5.423 `pod_char_traits.h` File Reference

### Classes

- struct [std::char\\_traits<\\_\\_gnu\\_cxx::character<\\_Value, \\_Int, \\_St>>](#)
- struct [\\_\\_gnu\\_cxx::character<\\_Value, \\_Int, \\_St>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Functions

- `template<typename _Value, typename _Int, typename _St>`  
`bool __gnu_cxx::operator<(const character<_Value, _Int, _St> &lhs, const character<_Value, _Int, _St> &rhs)`
- `template<typename _Value, typename _Int, typename _St>`  
`bool __gnu_cxx::operator==(const character<_Value, _Int, _St> &lhs, const character<_Value, _Int, _St> &rhs)`

### 5.423.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.424 `point_const_iterator.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator<Value\\_Type, Entry, Simple, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.424.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 5.425 point\_const\_iterator.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::detail::left\\_child\\_next\\_sibling\\_heap\\_node\\_point\\_const\\_iterator\\_< Node, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.425.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

## 5.426 point\_const\_iterator.hpp File Reference

### 5.426.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

- This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 5.427 point\_iterator.hpp File Reference

### 5.427.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.

This file is intended to be included inside a class definition, with PB\_DS\_CLASS\_C\_DEC defined to the name of the enclosing class.

## 5.428 `point_iterators.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference](#)
- class [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_TREE_CONST_IT_C_DEC`
- `#define PB_DS_TREE_CONST_ODIR_IT_C_DEC`
- `#define PB_DS_TREE_IT_C_DEC`
- `#define PB_DS_TREE_ODIR_IT_C_DEC`

### 5.428.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.429 `pointer.h` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::Invalid\\_type](#)
- class [\\_\\_gnu\\_cxx::Pointer\\_adapter< \\_Storage\\_policy >](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl< \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::Relative\\_pointer\\_impl< const \\_Tp >](#)
- class [\\_\\_gnu\\_cxx::Std\\_pointer\\_impl< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::Unqualified\\_type< \\_Tp >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define _CXX_POINTER_ARITH_OPERATOR_SET(INT_TYPE)`
- `#define _GCC_CXX_POINTER_COMPARISON_OPERATION_SET(OPERATOR)`

## Functions

- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator!= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _CharT, typename _Traits, typename _StoreT >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, int __rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (int __lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`

- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__`  
`__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__`  
`__rhs)`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool __gnu_cxx::operator>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)`
- `template<typename _Tp >`  
`bool __gnu_cxx::operator>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__`  
`rhs)`

#### 5.429.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

##### Author

Bob Walters

Provides reusable `_Pointer_adapter` for assisting in the development of custom pointer types that can be used with the standard containers via the `allocator::pointer` and `allocator::const_pointer` typedefs.

#### 5.430 `policy_access_fn_imps.hpp` File Reference

##### 5.430.1 Detailed Description

Contains an implementation class for a `binary_heap`.

#### 5.431 `policy_access_fn_imps.hpp` File Reference

##### 5.431.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

#### 5.432 `policy_access_fn_imps.hpp` File Reference

##### 5.432.1 Detailed Description

Contains implementations of `cc_ht_map_`'s policy access functions.

## 5.433 policy\_access\_fn\_imps.hpp File Reference

### 5.433.1 Detailed Description

Contains implementations of gp\_ht\_map\_'s policy access functions.

## 5.434 policy\_access\_fn\_imps.hpp File Reference

### 5.434.1 Detailed Description

Contains an implementation class for left\_child\_next\_sibling\_heap\_.

## 5.435 policy\_access\_fn\_imps.hpp File Reference

### 5.435.1 Detailed Description

Contains an implementation class for ov\_tree.

## 5.436 policy\_access\_fn\_imps.hpp File Reference

### 5.436.1 Detailed Description

Contains an implementation class for pat\_trie.

## 5.437 pool\_allocator.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc<\\_Tp>](#)
- class [\\_\\_gnu\\_cxx::\\_\\_pool\\_alloc\\_base](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Functions

- `template<typename _Tp>  
bool \_\_gnu\_cxx::operator!= (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`
- `template<typename _Tp>  
bool \_\_gnu\_cxx::operator== (const __pool_alloc<_Tp> &, const __pool_alloc<_Tp> &)`

#### 5.437.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

### 5.438 postypes.h File Reference

#### Classes

- class [std::fpos<\\_StateT>](#)

#### Namespaces

- [std](#)

#### Typedefs

- typedef long long [std::streamoff](#)
- typedef fpos< mbstate\_t > [std::streampos](#)
- typedef ptrdiff\_t [std::streamsize](#)
- typedef fpos< mbstate\_t > [std::u16streampos](#)
- typedef fpos< mbstate\_t > [std::u32streampos](#)
- typedef fpos< mbstate\_t > [std::wstreampos](#)

#### Functions

- template<typename \_StateT>  
bool **std::operator!=** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)
- template<typename \_StateT>  
bool **std::operator==** (const fpos< \_StateT > &\_\_lhs, const fpos< \_StateT > &\_\_rhs)

#### 5.438.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iosfwd>`.

### 5.439 predefined\_ops.h File Reference

#### Namespaces

- [\\_\\_gnu\\_cxx](#)



## Functions

- `template<typename _Compare >`  
`constexpr _Iter_comp_iter< _Compare > __gnu_cxx::__ops::__iter_comp_iter ( _Compare __comp)`
- `template<typename _Iterator >`  
`constexpr _Iter_equals_iter< _Iterator > __gnu_cxx::__ops::__iter_comp_iter ( _Iter_equal_to_iter, _Iterator __it)`
- `template<typename _Compare, typename _Iterator >`  
`constexpr _Iter_comp_to_iter< _Compare, _Iterator > __gnu_cxx::__ops::__iter_comp_iter ( _Iter_comp_↔  
_iter< _Compare > __comp, _Iterator __it)`
- `constexpr _Iter_less_val __gnu_cxx::__ops::__iter_comp_val ( _Iter_less_iter)`
- `constexpr _Iter_equal_to_val __gnu_cxx::__ops::__iter_comp_val ( _Iter_equal_to_iter)`
- `template<typename _Compare >`  
`constexpr _Iter_comp_val< _Compare > __gnu_cxx::__ops::__iter_comp_val ( _Compare __comp)`
- `template<typename _Compare >`  
`constexpr _Iter_comp_val< _Compare > __gnu_cxx::__ops::__iter_comp_val ( _Iter_comp_iter< _Compare  
> __comp)`
- `template<typename _Compare, typename _Value >`  
`_Iter_comp_to_val< _Compare, _Value > constexpr __gnu_cxx::__ops::__iter_comp_val ( _Compare __↔  
comp, _Value &__val)`
- `constexpr _Iter_equal_to_iter __gnu_cxx::__ops::__iter_equal_to_iter ()`
- `constexpr _Iter_equal_to_val __gnu_cxx::__ops::__iter_equal_to_val ()`
- `template<typename _Value >`  
`constexpr _Iter_equals_val< _Value > __gnu_cxx::__ops::__iter_equals_val ( _Value &__val)`
- `constexpr _Iter_less_iter __gnu_cxx::__ops::__iter_less_iter ()`
- `constexpr _Iter_less_val __gnu_cxx::__ops::__iter_less_val ()`
- `template<typename _Predicate >`  
`constexpr _Iter_negate< _Predicate > __gnu_cxx::__ops::__negate ( _Iter_pred< _Predicate > __pred)`
- `template<typename _Predicate >`  
`constexpr _Iter_pred< _Predicate > __gnu_cxx::__ops::__pred_iter ( _Predicate __pred)`
- `constexpr _Val_less_iter __gnu_cxx::__ops::__val_comp_iter ( _Iter_less_iter)`
- `template<typename _Compare >`  
`constexpr _Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter ( _Compare __comp)`
- `template<typename _Compare >`  
`constexpr _Val_comp_iter< _Compare > __gnu_cxx::__ops::__val_comp_iter ( _Iter_comp_iter< _Compare  
> __comp)`
- `constexpr _Val_less_iter __gnu_cxx::__ops::__val_less_iter ()`

## 5.439.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include `<algorithm>`.

## 5.440 prefix\_search\_node\_update\_imp.hpp File Reference

## 5.440.1 Detailed Description

Contains an implementation of `prefix_search_node_update`.

## 5.441 `priority_queue.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::priority\\_queue<\\_Tv, Cmp\\_Fn, Tag, \\_Alloc>](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.441.1 Detailed Description

Contains `priority_queues`.

## 5.442 `priority_queue_base_dispatch.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binary\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, pairing\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, rc\\_binomial\\_heap\\_tag, null\\_type>](#)
- struct [\\_\\_gnu\\_pbds::detail::container\\_base\\_dispatch<\\_VTp, Cmp\\_Fn, \\_Alloc, thin\\_heap\\_tag, null\\_type>](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_ASSERT_VALID(X)`
- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 5.442.1 Detailed Description

Contains an pqiative container dispatching base.

## 5.443 `probe_fn_base.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base<\\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.443.1 Detailed Description

Contains a probe policy base.

## 5.444 propagate\_const File Reference

## Classes

- class [std::experimental::fundamentals\\_v2::propagate\\_const<\\_Tp>](#)

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define \_GLIBCXX\_EXPERIMENTAL\_PROPAGATE\_CONST`

## Functions

- `template<typename _Tp>  
constexpr const _Tp & std::experimental::fundamentals\_v2::get\_underlying (const propagate_const<_Tp> &__pt) noexcept`
- `template<typename _Tp>  
constexpr _Tp & std::experimental::fundamentals\_v2::get\_underlying (propagate_const<_Tp> &__pt) noexcept`
- `template<typename _Tp>  
constexpr bool std::experimental::fundamentals\_v2::operator!= (const propagate_const<_Tp> &__pt, nullptr_t)`
- `template<typename _Tp>  
constexpr bool std::experimental::fundamentals\_v2::operator!= (nullptr_t, const propagate_const<_Tp> &__pu)`
- `template<typename _Tp, typename _Up>  
constexpr bool std::experimental::fundamentals\_v2::operator!= (const propagate_const<_Tp> &__pt, const propagate_const<_Up> &__pu)`
- `template<typename _Tp, typename _Up>  
constexpr bool std::experimental::fundamentals\_v2::operator!= (const propagate_const<_Tp> &__pt, const _Up &__u)`
- `template<typename _Tp, typename _Up>  
constexpr bool std::experimental::fundamentals\_v2::operator!= (const _Tp &__t, const propagate_const<_Up> &__pu)`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator< (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`  
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const propagate_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator<= (const _Tp &__t, const propagate_const<`  
`_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`nullptr_t)`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (nullptr_t, const propagate_const< _Tp >`  
`&__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const propagate_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator== (const _Tp &__t, const propagate_const<`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const< _Tp > &__pt, const`  
`propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const propagate_const< _Tp > &__pt, const`  
`_Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator> (const _Tp &__t, const propagate_const< ↵`  
`_Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const< _Tp > &__pt,`  
`const propagate_const< _Up > &__pu)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const propagate_const< _Tp > &__pt,`  
`const _Up &__u)`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v2::operator>= (const _Tp &__t, const propagate_const<`  
`_Up > &__pu)`
- `template<typename _Tp >`  
`constexpr void std::experimental::fundamentals_v2::swap (propagate_const< _Tp > &__pt, propagate_↵`  
`const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)`

## 5.444.1 Detailed Description

This is a TS C++ Library header.

## 5.445 ptr\_traits.h File Reference

## Classes

- struct [std::pointer\\_traits<\\_Ptr>](#)
- struct [std::pointer\\_traits<\\_Tp\\*>](#)

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Tp>`  
`using std::\_\_get\_first\_arg\_t = typename __get_first_arg<_Tp>::type`
- `template<typename _Tp>`  
`using std::\_\_make\_not\_void = typename conditional<is_void<_Tp>::value, __undefined, _Tp>::type`
- `template<typename _Ptr, typename _Tp>`  
`using std::\_\_ptr\_rebind = typename pointer_traits<_Ptr>::template rebind<_Tp>`
- `template<typename _Tp, typename _Up>`  
`using std::\_\_replace\_first\_arg\_t = typename __replace_first_arg<_Tp, _Up>::type`

## Functions

- `template<typename _Tp>`  
`constexpr _Tp * std::\_\_to\_address (_Tp * __ptr) noexcept`
- `template<typename _Ptr>`  
`constexpr std::pointer\_traits<\_Ptr>::element\_type * std::\_\_to\_address (const _Ptr & __ptr)`

## 5.445.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.446 quadratic\_probe\_fn\_imp.hpp File Reference

## 5.446.1 Detailed Description

Contains a probe policy implementation

## 5.447 queue File Reference

### Macros

- `#define _GLIBCXX_QUEUE`

### 5.447.1 Detailed Description

This is a Standard C++ Library header.

## 5.448 queue.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue<\\_Tp>](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Macros

- `#define _GLIBCXX_VOLATILE`

### 5.448.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

### 5.448.2 Macro Definition Documentation

#### 5.448.2.1 \_GLIBCXX\_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

Definition at line 40 of file queue.h.

## 5.449 quicksort.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)`
- `template<typename _RAIter, typename _Compare >`  
`std::iterator\_traits<_RAIter >::difference_type \_\_gnu\_parallel::\_\_parallel\_sort\_qs\_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator\_traits<_RAIter >::difference_type __pivot_rank, typename std::iterator\_traits<_RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)`

## 5.449.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

## 5.450 quoted\_string.h File Reference

## Classes

- `struct std::\_\_detail::Quoted\_string<_String, _CharT >`

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Functions

- `template<typename _CharT, typename _Traits >`  
`std::basic\_ostream<_CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream<_CharT, _Traits > & __os, const Quoted\_string< const _CharT *, _CharT > & __str)`
- `template<typename _CharT, typename _Traits, typename _String >`  
`std::basic\_ostream<_CharT, _Traits > & std::\_\_detail::operator<< (std::basic\_ostream<_CharT, _Traits > & __os, const Quoted\_string<_String, _CharT > & __str)`
- `template<typename _CharT, typename _Traits, typename _Alloc >`  
`std::basic\_istream<_CharT, _Traits > & std::\_\_detail::operator>> (std::basic\_istream<_CharT, _Traits > & __is, const Quoted\_string< basic\_string<_CharT, _Traits, _Alloc > &, _CharT > & __str)`

#### 5.450.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iomanip>`.

#### 5.451 `r_erase_fn_imps.hpp` File Reference

##### 5.451.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

#### 5.452 `r_erase_fn_imps.hpp` File Reference

##### 5.452.1 Detailed Description

Contains an implementation class for `pat_trie`.

#### 5.453 `random` File Reference

##### Macros

- `#define _GLIBCXX_RANDOM`

##### 5.453.1 Detailed Description

This is a Standard C++ Library header.

#### 5.454 `random` File Reference

##### Namespaces

- [std](#)
- [std::experimental](#)

##### Macros

- `#define __cpp_lib_experimental_randint`
- `#define _GLIBCXX_EXPERIMENTAL_RANDOM`



## Functions

- `std::default_random_engine` & `std::experimental::fundamentals_v2::S_randint_engine` ()
- `template<typename _IntType >`  
`_IntType std::experimental::fundamentals_v2::randint` (`_IntType __a`, `_IntType __b`)
- `void std::experimental::fundamentals_v2::reseed` ()
- `void std::experimental::fundamentals_v2::reseed` (`default_random_engine::result_type __value`)

## 5.454.1 Detailed Description

This is a TS C++ Library header.

## 5.455 random.h File Reference

## Classes

- class `std::bernoulli_distribution`
- class `std::binomial_distribution< _IntType >`
- class `std::cauchy_distribution< _RealType >`
- class `std::chi_squared_distribution< _RealType >`
- class `std::discard_block_engine< _RandomNumberEngine, __p, __r >`
- class `std::discrete_distribution< _IntType >`
- class `std::exponential_distribution< _RealType >`
- class `std::extreme_value_distribution< _RealType >`
- class `std::fisher_f_distribution< _RealType >`
- class `std::gamma_distribution< _RealType >`
- class `std::geometric_distribution< _IntType >`
- class `std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >`
- class `std::linear_congruential_engine< _UIntType, __a, __c, __m >`
- class `std::lognormal_distribution< _RealType >`
- class `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`
- class `std::negative_binomial_distribution< _IntType >`
- class `std::normal_distribution< _RealType >`
- struct `std::piecewise_linear_distribution< _RealType >::param_type`
- struct `std::piecewise_constant_distribution< _RealType >::param_type`
- struct `std::discrete_distribution< _IntType >::param_type`
- struct `std::extreme_value_distribution< _RealType >::param_type`
- struct `std::weibull_distribution< _RealType >::param_type`
- struct `std::exponential_distribution< _RealType >::param_type`
- struct `std::poisson_distribution< _IntType >::param_type`
- struct `std::negative_binomial_distribution< _IntType >::param_type`
- struct `std::geometric_distribution< _IntType >::param_type`
- struct `std::binomial_distribution< _IntType >::param_type`
- struct `std::bernoulli_distribution::param_type`
- struct `std::student_t_distribution< _RealType >::param_type`
- struct `std::fisher_f_distribution< _RealType >::param_type`
- struct `std::cauchy_distribution< _RealType >::param_type`
- struct `std::chi_squared_distribution< _RealType >::param_type`

- struct `std::gamma_distribution<_RealType>::param_type`
- struct `std::lognormal_distribution<_RealType>::param_type`
- struct `std::normal_distribution<_RealType>::param_type`
- struct `std::uniform_real_distribution<_RealType>::param_type`
- class `std::piecewise_constant_distribution<_RealType>`
- class `std::piecewise_linear_distribution<_RealType>`
- class `std::poisson_distribution<_IntType>`
- class `std::random_device`
- class `std::seed_seq`
- class `std::shuffle_order_engine<_RandomNumberEngine, __k>`
- class `std::student_t_distribution<_RealType>`
- class `std::subtract_with_carry_engine<_UIntType, __w, __s, __r>`
- class `std::uniform_real_distribution<_RealType>`
- class `std::weibull_distribution<_RealType>`

## Namespaces

- `std`
- `std::__detail`

## Typedefs

- typedef `minstd_rand0` **`std::default_random_engine`**
- typedef `shuffle_order_engine< minstd_rand0, 256 >` **`std::knuth_b`**
- typedef `linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL >` `std::minstd_rand`
- typedef `linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >` `std::minstd_rand0`
- typedef `mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL >` `std::mt19937`
- typedef `mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67ffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL >` `std::mt19937_64`
- typedef `discard_block_engine< ranlux24_base, 223, 23 >` **`std::ranlux24`**
- typedef `subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >` **`std::ranlux24_base`**
- typedef `discard_block_engine< ranlux48_base, 389, 11 >` **`std::ranlux48`**
- typedef `subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 >` **`std::ranlux48_base`**

## Functions

- template<typename `_RealType`, size\_t `__bits`, typename `_UniformRandomNumberGenerator`>  
`_RealType` `std::generate_canonical` (`_UniformRandomNumberGenerator` &`__g`)
- template<typename `_UIntType`, `_UIntType` `__a`, `_UIntType` `__c`, `_UIntType` `__m`>  
bool `std::operator!=` (const `std::linear_congruential_engine<_UIntType, __a, __c, __m>` &`__lhs`, const `std::linear_congruential_engine<_UIntType, __a, __c, __m>` &`__rhs`)
- template<typename `_UIntType`, size\_t `__w`, size\_t `__n`, size\_t `__m`, size\_t `__r`, `_UIntType` `__a`, size\_t `__u`, `_UIntType` `__d`, size\_t `__s`, `_UIntType` `__b`, size\_t `__t`, `_UIntType` `__c`, size\_t `__l`, `_UIntType` `__f`>  
bool `std::operator!=` (const `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>` &`__lhs`, const `std::mersenne_twister_engine<_UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f>` &`__rhs`)

- `template<typename _UIntType, size_t __w, size_t __s, size_t __r>`  
`bool std::operator!= (const std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__lhs, const`  
`std::subtract_with_carry_engine< _UIntType, __w, __s, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r>`  
`bool std::operator!= (const std::discard_block_engine< _RandomNumberEngine, __p, __r > &__lhs, const`  
`std::discard_block_engine< _RandomNumberEngine, __p, __r > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType >`  
`bool std::operator!= (const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__lhs,`  
`const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__rhs)`
- `template<typename _RandomNumberEngine, size_t __k>`  
`bool std::operator!= (const std::shuffle_order_engine< _RandomNumberEngine, __k > &__lhs, const`  
`std::shuffle_order_engine< _RandomNumberEngine, __k > &__rhs)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_int_distribution< _IntType > &__d1, const std::uniform_int_distribution<`  
`_IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::uniform_real_distribution< _IntType > &__d1, const std::uniform_real_distribution<`  
`_IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _`  
`RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::lognormal_distribution< _RealType > &__d1, const std::lognormal_distribution<`  
`_RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::gamma_distribution< _RealType > &__d1, const std::gamma_distribution< _`  
`RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::chi_squared_distribution< _RealType > &__d1, const std::chi_squared_distribution<`  
`_RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::cauchy_distribution< _RealType > &__d1, const std::cauchy_distribution< _`  
`RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::fisher_f_distribution< _RealType > &__d1, const std::fisher_f_distribution< _`  
`RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::student_t_distribution< _RealType > &__d1, const std::student_t_distribution<`  
`_RealType > &__d2)`
- `bool std::operator!= (const std::bernoulli_distribution &__d1, const std::bernoulli_distribution &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::binomial_distribution< _IntType > &__d1, const std::binomial_distribution< _`  
`IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::geometric_distribution< _IntType > &__d1, const std::geometric_distribution<`  
`_IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::negative_binomial_distribution< _IntType > &__d1, const std::negative_binomial_distribution<`  
`_IntType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::poisson_distribution< _IntType > &__d1, const std::poisson_distribution< _Int`  
`Type > &__d2)`

- `template<typename _RealType >`  
`bool std::operator!= (const std::exponential_distribution< _RealType > &__d1, const std::exponential_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::weibull_distribution< _RealType > &__d1, const std::weibull_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::extreme_value_distribution< _RealType > &__d1, const std::extreme_value_distribution< _RealType > &__d2)`
- `template<typename _IntType >`  
`bool std::operator!= (const std::discrete_distribution< _IntType > &__d1, const std::discrete_distribution< _IntType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_constant_distribution< _RealType > &__d1, const std::piecewise_constant_distribution< _RealType > &__d2)`
- `template<typename _RealType >`  
`bool std::operator!= (const std::piecewise_linear_distribution< _RealType > &__d1, const std::piecewise_linear_distribution< _RealType > &__d2)`
- `template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`

- `template<typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >  
std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,  
std::extreme_value_distribution< _RealType > &__x)`

### 5.455.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 5.456 random.tcc File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Macros

- `#define _RANDOM_TCC`

### Functions

- `template<typename _ValT, typename _CharT, typename _Traits >  
basic_istream< _CharT, _Traits > & std::__detail::__extract_params (basic_istream< _CharT, _Traits > &__is,  
vector< _ValT > &__vals, size_t __n)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >  
_OutputIterator std::__detail::__normalize (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
const _Tp &__factor)`
- `template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator >  
_RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >  
std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,  
const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`

- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f`  
`> &__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const negative_binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_int_distribution< _IntType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const`  
`std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const gamma_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits>`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_constant_distribution< _RealType > &__x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os,`  
`const piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::bernoulli_distribution &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const`  
`std::extreme_value_distribution< _RealType > &__x)`
- `template<typename _RealType >`  
`bool std::operator== (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)`
- `template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)`
- `template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b,`  
`size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`  
`&__x)`
- `template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`discard_block_engine< _RandomNumberEngine, __p, __r > &__x)`
- `template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`shuffle_order_engine< _RandomNumberEngine, __k > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`negative_binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`poisson_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,`  
`binomial_distribution< _IntType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__,`  
`std::uniform_int_distribution< _IntType > &)`



- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)`
- `template<typename _IntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)`

#### 5.456.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.



## 5.457 random.tcc File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Macros

- `#define _EXT_RANDOM_TCC`

## Functions

- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::beta_distribution< _RealType > & __x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const rice_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const nakagami_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const pareto_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const k_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const arcsine_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const hoyt_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::triangular_distribution< _RealType > & __x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::von_mises_distribution< _RealType > & __x)`
- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > & __os, const __gnu_cxx::hypergeometric_distribution< _UIntType > & __x)`

- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &↵`  
`__os, const logistic_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &↵`  
`__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_ostream< _CharT, _Traits > & gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &↵`  
`__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>`  
`bool gnu_cxx::operator== (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __↵`  
`pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4`  
`> &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __↵`  
`sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)`
- `template<size_t _Dimen, typename _RealType >`  
`bool gnu_cxx::operator== (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const`  
`__gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)`
- `template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t`  
`__msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT`  
`, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2,`  
`__msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::beta_distribution< _RealType > &__x)`
- `template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, rice_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, nakagami_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, pareto_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, k_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, arcsine_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, hoyt_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::triangular_distribution< _RealType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↵`  
`__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)`

- `template<typename _UIntType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↔`  
`__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)`
- `template<typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↔`  
`__is, logistic_distribution< _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↔`  
`__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)`
- `template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits >`  
`std::basic_istream< _CharT, _Traits > & __gnu_cxx::operator>> (std::basic_istream< _CharT, _Traits > &↔`  
`__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)`

#### 5.457.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/random>`.

## 5.458 random\_number.h File Reference

### Classes

- class [\\_\\_gnu\\_parallel::\\_RandomNumber](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 5.458.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

## 5.459 random\_shuffle.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_DRandomShufflingGlobalData< \\_RAIter >](#)
- struct [\\_\\_gnu\\_parallel::\\_DRSSorterPU< \\_RAIter, \\_RandomNumberGenerator >](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

## Typedefs

- typedef unsigned short [\\_\\_gnu\\_parallel::\\_\\_BinIndex](#)

## Functions

- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator \_\_rng= \_RandomNumber())
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs](#) (\_RAIter \_\_begin, \_RAIter \_\_end, typename [std::iterator\\_traits](#)<\_RAIter >::difference\_type \_\_n, \_ThreadIndex \_\_num\_threads, \_RandomNumberGenerator &\_\_rng)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu](#) (\_DRSSorterPU< \_RAIter, \_RandomNumberGenerator > \* \_\_pus)
- template<typename \_RandomNumberGenerator >  
int [\\_\\_gnu\\_parallel::\\_\\_random\\_number\\_pow2](#) (int \_\_logp, \_RandomNumberGenerator &\_\_rng)
- template<typename \_Tp >  
\_Tp [\\_\\_gnu\\_parallel::\\_\\_round\\_up\\_to\\_pow2](#) (\_Tp \_\_x)
- template<typename \_RAIter, typename \_RandomNumberGenerator >  
void [\\_\\_gnu\\_parallel::\\_\\_sequential\\_random\\_shuffle](#) (\_RAIter \_\_begin, \_RAIter \_\_end, \_RandomNumberGenerator &\_\_rng)

### 5.459.1 Detailed Description

Parallel implementation of `std::random_shuffle()`. This file is a GNU parallel extension to the Standard C++ Library.

### 5.460 range\_access.h File Reference

## Classes

- class [std::valarray](#)< \_Tp >

## Namespaces

- [std](#)

## Functions

- `template<typename _Container >`  
`constexpr auto std::begin (_Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Container >`  
`constexpr auto std::begin (const _Container &__cont) -> decltype(__cont.begin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::begin (_Tp(&__arr)[_Nm]) noexcept`
- `template<class _Tp >`  
`_Tp * std::begin (valarray< _Tp > &__va)`
- `template<class _Tp >`  
`const _Tp * std::begin (const valarray< _Tp > &__va)`
- `template<typename _Container >`  
`constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> de-`  
`cltype(std::begin(__cont))`
- `template<typename _Container >`  
`constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(←`  
`__cont))`
- `template<typename _Container >`  
`constexpr auto std::crbegin (const _Container &__cont) -> decltype(std::rbegin(__cont))`
- `template<typename _Container >`  
`constexpr auto std::crend (const _Container &__cont) -> decltype(std::rend(__cont))`
- `template<typename _Container >`  
`constexpr auto std::end (_Container &__cont) -> decltype(__cont.end())`
- `template<typename _Container >`  
`constexpr auto std::end (const _Container &__cont) -> decltype(__cont.end())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr _Tp * std::end (_Tp(&__arr)[_Nm]) noexcept`
- `template<class _Tp >`  
`_Tp * std::end (valarray< _Tp > &__va)`
- `template<class _Tp >`  
`const _Tp * std::end (const valarray< _Tp > &__va)`
- `template<typename _Container >`  
`constexpr auto std::rbegin (_Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Container >`  
`constexpr auto std::rbegin (const _Container &__cont) -> decltype(__cont.rbegin())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Tp >`  
`constexpr reverse_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il) noexcept`
- `template<typename _Container >`  
`constexpr auto std::rend (_Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Container >`  
`constexpr auto std::rend (const _Container &__cont) -> decltype(__cont.rend())`
- `template<typename _Tp, size_t _Nm>`  
`constexpr reverse_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm]) noexcept`
- `template<typename _Tp >`  
`constexpr reverse_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il) noexcept`

## 5.460.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.461 range\_cmp.h File Reference

### 5.461.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.462 ranged\_hash\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, Store\\_Hash >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.462.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

## 5.463 ranged\_probe\_fn.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, Store\\_Hash >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >](#)
- class [\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.463.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probing.

## 5.464 ranges File Reference

## Macros

- `#define _GLIBCXX_RANGES`

### 5.464.1 Detailed Description

This is a Standard C++ Library header.

## 5.465 ranges\_algo.h File Reference

### 5.465.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.466 ranges\_algobase.h File Reference

### 5.466.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.467 `ranges_uninitialized.h` File Reference

### 5.467.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.468 `ratio` File Reference

### Classes

- struct [std::ratio<\\_Num, \\_Den >](#)
- struct [std::ratio\\_equal<\\_R1, \\_R2 >](#)
- struct [std::ratio\\_greater<\\_R1, \\_R2 >](#)
- struct [std::ratio\\_greater\\_equal<\\_R1, \\_R2 >](#)
- struct [std::ratio\\_less<\\_R1, \\_R2 >](#)
- struct [std::ratio\\_less\\_equal<\\_R1, \\_R2 >](#)
- struct [std::ratio\\_not\\_equal<\\_R1, \\_R2 >](#)

### Namespaces

- [std](#)

### Macros

- `#define \_GLIBCXX\_RATIO`

### Typedefs

- `typedef ratio< 1, 1000000000000000000 > std::atto`
- `typedef ratio< 1, 100 > std::centi`
- `typedef ratio< 10, 1 > std::deca`
- `typedef ratio< 1, 10 > std::deci`
- `typedef ratio< 1000000000000000000, 1 > std::exa`
- `typedef ratio< 1, 1000000000000000 > std::femto`
- `typedef ratio< 1000000000, 1 > std::giga`
- `typedef ratio< 100, 1 > std::hecto`
- `typedef ratio< 1000, 1 > std::kilo`
- `typedef ratio< 1000000, 1 > std::mega`
- `typedef ratio< 1, 1000000 > std::micro`
- `typedef ratio< 1, 1000 > std::milli`
- `typedef ratio< 1, 1000000000 > std::nano`
- `typedef ratio< 1000000000000000000, 1 > std::peta`
- `typedef ratio< 1, 1000000000000 > std::pico`
- `template<typename _R1, typename _R2 >`  
`using std::ratio\_add = typename \_\_ratio\_add<_R1, _R2>::type`
- `template<typename _R1, typename _R2 >`  
`using std::ratio\_divide = typename \_\_ratio\_divide<_R1, _R2>::type`
- `template<typename _R1, typename _R2 >`  
`using std::ratio\_multiply = typename \_\_ratio\_multiply<_R1, _R2>::type`
- `template<typename _R1, typename _R2 >`  
`using std::ratio\_subtract = typename \_\_ratio\_subtract<_R1, _R2>::type`
- `typedef ratio< 1000000000000, 1 > std::tera`



### 5.468.1 Detailed Description

This is a Standard C++ Library header.

## 5.469 ratio File Reference

### Namespaces

- [std](#)
- [std::tr2](#)

### 5.469.1 Detailed Description

This is a TR2 C++ Library header.

## 5.470 ratio File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_RATIO`

### Variables

- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_greater_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_greater_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_less_equal_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_less_v`
- `template<typename _R1, typename _R2 >`  
`constexpr bool std::experimental::fundamentals_v1::ratio_not_equal_v`

### 5.470.1 Detailed Description

This is a TS C++ Library header.

## 5.471 `rb_tree` File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::rb\\_tree< \\_Key, \\_Value, \\_KeyOfValue, \\_Compare, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### Macros

- `#define _RB_TREE`

#### 5.471.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.472 `rb_tree.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RB_TREE_BASE`
- `#define PB_DS_RB_TREE_BASE_NAME`
- `#define PB_DS_RB_TREE_NAME`
- `#define PB_DS_STRUCT_ONLY_ASSERT_VALID(X)`

#### 5.472.1 Detailed Description

Contains an implementation for Red Black trees.

## 5.473 rc.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.473.1 Detailed Description

Contains a redundant (binary counter).

## 5.474 rc\_binomial\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_RC_C_DEC`

#### 5.474.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

## 5.475 rc\_string\_base.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base< \\_CharT, \\_Traits, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.475.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.476 `refwrap.h` File Reference

### Classes

- class [std::reference\\_wrapper<\\_Tp>](#)

## Namespaces

- [std](#)

### 5.476.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.477 `regex` File Reference

### Macros

- `#define _GLIBCXX_REGEX`

### 5.477.1 Detailed Description

This is a Standard C++ Library header.

## 5.478 `regex` File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_REGEX`

### 5.478.1 Detailed Description

This is a TS C++ Library header.

## 5.479 regex.h File Reference

### Classes

- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`
- class `std::basic_regex<_Ch_type, _Rx_traits >`
- class `std::basic_regex<_Ch_type, _Rx_traits >`
- class `std::match_results<_Bi_iter, _Alloc >`
- class `std::match_results<_Bi_iter, _Alloc >`
- class `std::regex_iterator<_Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_token_iterator<_Bi_iter, _Ch_type, _Rx_traits >`
- class `std::regex_traits<_Ch_type >`
- class `std::sub_match<_Bilter >`

### Namespaces

- `std`
- `std::__detail`

### Typedefs

- typedef `match_results< const char * >` **`std::cmatch`**
- typedef `regex_iterator< const char * >` **`std::cregex_iterator`**
- typedef `regex_token_iterator< const char * >` `std::cregex_token_iterator`
- typedef `sub_match< const char * >` `std::csub_match`
- typedef `basic_regex< char >` `std::regex`
- typedef `match_results< string::const_iterator >` **`std::smatch`**
- typedef `regex_iterator< string::const_iterator >` **`std::sregex_iterator`**
- typedef `regex_token_iterator< string::const_iterator >` `std::sregex_token_iterator`
- typedef `sub_match< string::const_iterator >` `std::ssub_match`
- typedef `match_results< const wchar_t * >` **`std::wcmatch`**
- typedef `regex_iterator< const wchar_t * >` **`std::wcregex_iterator`**
- typedef `regex_token_iterator< const wchar_t * >` `std::wcregex_token_iterator`
- typedef `sub_match< const wchar_t * >` `std::wcs_sub_match`
- typedef `basic_regex< wchar_t >` `std::wregex`
- typedef `match_results< wstring::const_iterator >` **`std::wsmatch`**
- typedef `regex_iterator< wstring::const_iterator >` **`std::wsregex_iterator`**
- typedef `regex_token_iterator< wstring::const_iterator >` `std::wsregex_token_iterator`
- typedef `sub_match< wstring::const_iterator >` `std::wssub_match`

## Enumerations

- enum **\_RegexExecutorPolicy** : int { **\_S\_auto**, **\_S\_alternate** }

## Functions

- template<typename \_Bilter, typename \_Alloc, typename \_CharT, typename \_TraitsT, \_RegexExecutorPolicy \_\_policy, bool \_\_match\_↵ mode>  
bool **std::detail::regex\_algo\_impl** ( \_Bilter \_\_s, \_Bilter \_\_e, match\_results< \_Bilter, \_Alloc > &\_\_m, const basic\_regex< \_CharT, \_TraitsT > &\_\_re, regex\_constants::match\_flag\_type \_\_flags)
- template<typename \_Bi\_iter, class \_Alloc >  
bool **std::operator!=** (const match\_results< \_Bi\_iter, \_Alloc > &\_\_m1, const match\_results< \_Bi\_iter, \_Alloc > &\_\_m2)
- template<typename \_Bi\_iter, typename \_Alloc >  
bool **std::operator==** (const match\_results< \_Bi\_iter, \_Alloc > &\_\_m1, const match\_results< \_Bi\_iter, \_Alloc > &\_\_m2)
- template<typename \_Bi\_iter, typename \_Alloc >  
void **std::swap** (match\_results< \_Bi\_iter, \_Alloc > &\_\_lhs, match\_results< \_Bi\_iter, \_Alloc > &\_\_rhs) noexcept

## Matching, Searching, and Replacing

- template<typename \_Bi\_iter, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_match** ( \_Bi\_iter \_\_s, \_Bi\_iter \_\_e, match\_results< \_Bi\_iter, \_Alloc > &\_\_m, const basic\_↵ regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_↵ \_default)
- template<typename \_Bi\_iter, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_match** ( \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_default)
- template<typename \_Ch\_type, typename \_Alloc, typename \_Rx\_traits >  
bool **std::regex\_match** (const \_Ch\_type \*\_\_s, match\_results< const \_Ch\_type \*, \_Alloc > &\_\_m, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_f=regex\_constants\_↵ ::match\_default)
- template<typename \_Ch\_traits, typename \_Ch\_alloc, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_match** (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &\_\_s, match\_results< type\_↵ name basic\_string< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc >::const\_iterator, \_Alloc > &\_\_m, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_default)
- template<typename \_Ch\_traits, typename \_Ch\_alloc, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_match** (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc > &&, match\_results< typename basic\_string< \_Ch\_type, \_Ch\_traits, \_Ch\_alloc >::const\_iterator, \_Alloc > &, const basic\_regex< \_Ch\_type, \_Rx\_traits > &, regex\_constants::match\_flag\_type=regex\_constants::match\_default)=delete
- template<typename \_Ch\_type, class \_Rx\_traits >  
bool **std::regex\_match** (const \_Ch\_type \*\_\_s, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_↵ constants::match\_flag\_type \_\_f=regex\_constants::match\_default)
- template<typename \_Ch\_traits, typename \_Str\_allocator, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_match** (const basic\_string< \_Ch\_type, \_Ch\_traits, \_Str\_allocator > &\_\_s, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_default)
- template<typename \_Bi\_iter, typename \_Alloc, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_search** ( \_Bi\_iter \_\_s, \_Bi\_iter \_\_e, match\_results< \_Bi\_iter, \_Alloc > &\_\_m, const basic\_↵ regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_↵ \_default)
- template<typename \_Bi\_iter, typename \_Ch\_type, typename \_Rx\_traits >  
bool **std::regex\_search** ( \_Bi\_iter \_\_first, \_Bi\_iter \_\_last, const basic\_regex< \_Ch\_type, \_Rx\_traits > &\_\_re, regex\_constants::match\_flag\_type \_\_flags=regex\_constants::match\_default)

- `template<typename _Ch_type, class _Alloc, class _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const`  
`basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::`  
`::match_default)`
- `template<typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_`  
`constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_`  
`regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_`  
`default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-`  
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<`  
`_Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)`
- `template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits >`  
`bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< type-`  
`name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _`  
`Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type`  
`__flags=regex_constants::match_default)`
- `template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type >`  
`_Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,`  
`_Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::`  
`::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__`  
`s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt,`  
`regex_constants::match_flag_type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s,`  
`const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_`  
`type __flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa >`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __`  
`flags=regex_constants::match_default)`
- `template<typename _Rx_traits, typename _Ch_type >`  
`basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_`  
`traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_`  
`default)`

### 5.479.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.480 regex.tcc File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### 5.480.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.481 `regex_automaton.h` File Reference

#### Classes

- class `std::__detail::_StateSeq<_TraitsT>`

#### Namespaces

- `std`
- `std::__detail`

#### Macros

- `#define _GLIBCXX_REGEX_STATE_LIMIT`

#### Typedefs

- `template<typename _CharT>`  
using `std::__detail::_Matcher` = `std::function< bool(_CharT)>`
- `typedef long std::__detail::_StatelIdT`

#### Enumerations

- `enum std::__detail::_Opcode : int {`  
    `_S_opcode_unknown, _S_opcode_alternative, _S_opcode_repeat, _S_opcode_backref,`  
    `_S_opcode_line_begin_assertion, _S_opcode_line_end_assertion, _S_opcode_word_boundary, _S_↵`  
    `opcode_subexpr_lookahead,`  
    `_S_opcode_subexpr_begin, _S_opcode_subexpr_end, _S_opcode_dummy, _S_opcode_match,`  
    `_S_opcode_accept }`

#### Variables

- `static const _StatelIdT std::__detail::_S_invalid_state_id`

#### 5.481.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.



## 5.482 regex\_automaton.tcc File Reference

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## 5.482.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.483 regex\_compiler.h File Reference

## Classes

- struct [std::\\_\\_detail::BracketMatcher<\\_TraitsT, \\_\\_icase, \\_\\_collate>](#) >
- struct [std::\\_\\_detail::BracketMatcher<\\_TraitsT, \\_\\_icase, \\_\\_collate>](#) >
- class [std::\\_\\_detail::\\_Compiler<\\_TraitsT>](#) >
- class [std::regex\\_traits<\\_Ch\\_type>](#) >

## Namespaces

- [std](#)
- [std::\\_\\_detail](#)

## Typedefs

- template<typename \_Iter, typename \_TraitsT>  
using [std::\\_\\_detail::\\_\\_disable\\_if\\_contiguous\\_iter](#) = \_\_enable\_if\_t< !\_\_is\_contiguous\_iter<\_Iter>::value, [std::shared\\_ptr](#)< const \_NFA<\_TraitsT> > > >
- template<typename \_Iter, typename \_TraitsT>  
using [std::\\_\\_detail::\\_\\_enable\\_if\\_contiguous\\_iter](#) = \_\_enable\_if\_t< \_\_is\_contiguous\_iter<\_Iter>::value, [std::shared\\_ptr](#)< const \_NFA<\_TraitsT> > > >

## Functions

- template<typename \_TraitsT, typename \_FwdIter>  
[\\_\\_enable\\_if\\_contiguous\\_iter<\\_FwdIter, \\_TraitsT>](#) [std::\\_\\_detail::\\_\\_compile\\_nfa](#) (\_FwdIter \_\_first, \_FwdIter ↵ \_\_last, const typename \_TraitsT::locale\_type & \_\_loc, regex\_constants::syntax\_option\_type \_\_flags)
- template<typename \_TraitsT, typename \_FwdIter>  
[\\_\\_disable\\_if\\_contiguous\\_iter<\\_FwdIter, \\_TraitsT>](#) [std::\\_\\_detail::\\_\\_compile\\_nfa](#) (\_FwdIter \_\_first, \_FwdIter \_\_last, const typename \_TraitsT::locale\_type & \_\_loc, regex\_constants::syntax\_option\_type \_\_flags)

#### 5.483.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.484 `regex_compiler.tcc` File Reference

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### Macros

- `#define __INSERT_REGEX_MATCHER(__func, ...)`

#### 5.484.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

### 5.485 `regex_constants.h` File Reference

#### Namespaces

- [std](#)
- [std::regex\\_constants](#)

#### 5.1 Regular Expression Syntax Options

- enum [std::regex\\_constants::\\_\\_syntax\\_option](#) {  
    [\\_S\\_icode](#), [\\_S\\_nosubs](#), [\\_S\\_optimize](#), [\\_S\\_collate](#),  
    [\\_S\\_ECMAScript](#), [\\_S\\_basic](#), [\\_S\\_extended](#), [\\_S\\_awk](#),  
    [\\_S\\_grep](#), [\\_S\\_egrep](#), [\\_S\\_polynomial](#), [\\_S\\_syntax\\_last](#) }
- enum [std::regex\\_constants::syntax\\_option\\_type](#) : unsigned int
- constexpr syntax\_option\_type [std::regex\\_constants::icase](#)
- constexpr syntax\_option\_type [std::regex\\_constants::nosubs](#)
- constexpr syntax\_option\_type [std::regex\\_constants::optimize](#)
- constexpr syntax\_option\_type [std::regex\\_constants::collate](#)
- constexpr syntax\_option\_type [std::regex\\_constants::ECMAScript](#)
- constexpr syntax\_option\_type [std::regex\\_constants::basic](#)
- constexpr syntax\_option\_type [std::regex\\_constants::extended](#)
- constexpr syntax\_option\_type [std::regex\\_constants::awk](#)
- constexpr syntax\_option\_type [std::regex\\_constants::grep](#)
- constexpr syntax\_option\_type [std::regex\\_constants::egrep](#)

- constexpr syntax\_option\_type [std::regex\\_constants::\\_\\_polynomial](#)
- constexpr syntax\_option\_type [std::regex\\_constants::operator&](#) (syntax\_option\_type \_\_a, syntax\_option\_type \_\_b)
- constexpr syntax\_option\_type [std::regex\\_constants::operator|](#) (syntax\_option\_type \_\_a, syntax\_option\_type \_\_b)
- constexpr syntax\_option\_type [std::regex\\_constants::operator^](#) (syntax\_option\_type \_\_a, syntax\_option\_type \_\_b)
- constexpr syntax\_option\_type [std::regex\\_constants::operator~](#) (syntax\_option\_type \_\_a)
- syntax\_option\_type & [std::regex\\_constants::operator&=](#) (syntax\_option\_type &\_\_a, syntax\_option\_type \_\_b)
- syntax\_option\_type & [std::regex\\_constants::operator|=](#) (syntax\_option\_type &\_\_a, syntax\_option\_type \_\_b)
- syntax\_option\_type & [std::regex\\_constants::operator^=](#) (syntax\_option\_type &\_\_a, syntax\_option\_type \_\_b)

## 5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum [std::regex\\_constants::\\_\\_match\\_flag](#) {  
[\\_S\\_not\\_bol](#), [\\_S\\_not\\_eol](#), [\\_S\\_not\\_bow](#), [\\_S\\_not\\_eow](#),  
[\\_S\\_any](#), [\\_S\\_not\\_null](#), [\\_S\\_continuous](#), [\\_S\\_prev\\_avail](#),  
[\\_S\\_sed](#), [\\_S\\_no\\_copy](#), [\\_S\\_first\\_only](#), [\\_S\\_match\\_flag\\_last](#) }
- enum [std::regex\\_constants::match\\_flag\\_type](#) : unsigned int
- constexpr match\_flag\_type [std::regex\\_constants::match\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eol](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_bow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_eow](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_any](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_not\\_null](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_continuous](#)
- constexpr match\_flag\_type [std::regex\\_constants::match\\_prev\\_avail](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_default](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_sed](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_no\\_copy](#)
- constexpr match\_flag\_type [std::regex\\_constants::format\\_first\\_only](#)
- constexpr match\_flag\_type [std::regex\\_constants::operator&](#) (match\_flag\_type \_\_a, match\_flag\_type \_\_b)
- constexpr match\_flag\_type [std::regex\\_constants::operator|](#) (match\_flag\_type \_\_a, match\_flag\_type \_\_b)
- constexpr match\_flag\_type [std::regex\\_constants::operator^](#) (match\_flag\_type \_\_a, match\_flag\_type \_\_b)
- constexpr match\_flag\_type [std::regex\\_constants::operator~](#) (match\_flag\_type \_\_a)
- match\_flag\_type & [std::regex\\_constants::operator&=](#) (match\_flag\_type &\_\_a, match\_flag\_type \_\_b)
- match\_flag\_type & [std::regex\\_constants::operator|=](#) (match\_flag\_type &\_\_a, match\_flag\_type \_\_b)
- match\_flag\_type & [std::regex\\_constants::operator^=](#) (match\_flag\_type &\_\_a, match\_flag\_type \_\_b)

### 5.485.1 Detailed Description

Constant definitions for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.486 regex\_error.h File Reference

### Classes

- class [std::regex\\_error](#)

### Namespaces

- [std](#)
- [std::regex\\_constants](#)

### Functions

- void **std::\_\_throw\_regex\_error** (regex\_constants::error\_type \_\_ecode)
- void **std::\_\_throw\_regex\_error** (regex\_constants::error\_type \_\_ecode, const char \*\_\_what)

## 5.3 Error Types

- enum [std::regex\\_constants::error\\_type](#) {  
    \_S\_error\_collate, \_S\_error\_ctype, \_S\_error\_escape, \_S\_error\_backref,  
    \_S\_error\_brack, \_S\_error\_paren, \_S\_error\_brace, \_S\_error\_badbrace,  
    \_S\_error\_range, \_S\_error\_space, \_S\_error\_badrepeat, \_S\_error\_complexity,  
    \_S\_error\_stack }
- constexpr error\_type [std::regex\\_constants::error\\_collate](#) (\_S\_error\_collate)
- constexpr error\_type [std::regex\\_constants::error\\_ctype](#) (\_S\_error\_ctype)
- constexpr error\_type [std::regex\\_constants::error\\_escape](#) (\_S\_error\_escape)
- constexpr error\_type [std::regex\\_constants::error\\_backref](#) (\_S\_error\_backref)
- constexpr error\_type [std::regex\\_constants::error\\_brack](#) (\_S\_error\_brack)
- constexpr error\_type [std::regex\\_constants::error\\_paren](#) (\_S\_error\_paren)
- constexpr error\_type [std::regex\\_constants::error\\_brace](#) (\_S\_error\_brace)
- constexpr error\_type [std::regex\\_constants::error\\_badbrace](#) (\_S\_error\_badbrace)
- constexpr error\_type [std::regex\\_constants::error\\_range](#) (\_S\_error\_range)
- constexpr error\_type [std::regex\\_constants::error\\_space](#) (\_S\_error\_space)
- constexpr error\_type [std::regex\\_constants::error\\_badrepeat](#) (\_S\_error\_badrepeat)
- constexpr error\_type [std::regex\\_constants::error\\_complexity](#) (\_S\_error\_complexity)
- constexpr error\_type [std::regex\\_constants::error\\_stack](#) (\_S\_error\_stack)

### 5.486.1 Detailed Description

Error and exception objects for the std regex library.

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.487 `regex_executor.h` File Reference

### Classes

- class `std::__detail::_Executor<_Bilter, _Alloc, _TraitsT, __dfs_mode >`

### Namespaces

- `std`
- `std::__detail`

#### 5.487.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.488 `regex_executor.tcc` File Reference

### Namespaces

- `std`
- `std::__detail`

#### 5.488.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.489 `regex_scanner.h` File Reference

### Classes

- class `std::__detail::_Scanner<_CharT >`

### Namespaces

- `std`
- `std::__detail`

#### 5.489.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.490 `regex_scanner.tcc` File Reference

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### 5.490.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<regex>`.

## 5.491 `resize_fn_imps.hpp` File Reference

### 5.491.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions.

## 5.492 `resize_fn_imps.hpp` File Reference

### 5.492.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions.

## 5.493 `resize_no_store_hash_fn_imps.hpp` File Reference

### 5.493.1 Detailed Description

Contains implementations of `cc_ht_map_`'s resize related functions, when the hash value is not stored.

## 5.494 `resize_no_store_hash_fn_imps.hpp` File Reference

### 5.494.1 Detailed Description

Contains implementations of `gp_ht_map_`'s resize related functions, when the hash value is not stored.

## 5.495 `resize_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::resize\\_policy<\\_Tp>](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.495.1 Detailed Description

Contains an implementation class for a `binary_heap`.

5.496 `resize_store_hash_fn_imps.hpp` File Reference

## 5.496.1 Detailed Description

Contains implementations of `cc_ht_map_`'s `resize` related functions, when the hash value is stored.

5.497 `resize_store_hash_fn_imps.hpp` File Reference

## 5.497.1 Detailed Description

Contains implementations of `gp_ht_map_`'s `resize` related functions, when the hash value is stored.

5.498 `rope` File Reference

## Classes

- class [\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#)
- class [\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::\\_\\_detail](#)
- [std](#)
- [std::tr1](#)

## Macros

- `#define __GC_CONST`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOC(_Tp, __name)`
- `#define __ROPE_DEFINE_ALLOCS(__a)`
- `#define __STATIC_IF_SGI_ALLOC`
- `#define __STL_FREE_STRING(__s, __l, __a)`
- `#define __STL_ROPE_FROM_UNOWNED_CHAR_PTR(__s, __size, __a)`
- `#define _ROPE`

## Typedefs

- typedef rope< char > **\_\_gnu\_cxx::crope**
- typedef rope< wchar\_t > **\_\_gnu\_cxx::wrope**

## Enumerations

- enum { **\_S\_max\_rope\_depth** }
- enum **\_Tag** { **\_S\_leaf**, **\_S\_concat**, **\_S\_substringfn**, **\_S\_function** }

## Functions

- crope::reference **\_\_gnu\_cxx::\_mutable\_reference\_at** (crope &\_\_c, std::size\_t \_\_i)
- template<typename \_ForwardIterator, typename \_Allocator >  
void **\_\_gnu\_cxx::\_Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, \_Allocator \_\_alloc)
- template<typename \_ForwardIterator, typename \_Tp >  
void **\_\_gnu\_cxx::\_Destroy\_const** (\_ForwardIterator \_\_first, \_ForwardIterator \_\_last, [std::allocator](#)< \_Tp >)
- template<class \_CharT >  
void **\_\_gnu\_cxx::\_S\_cond\_store\_eos** (\_CharT &)
- void **\_\_gnu\_cxx::\_S\_cond\_store\_eos** (char &\_\_c)
- void **\_\_gnu\_cxx::\_S\_cond\_store\_eos** (wchar\_t &\_\_c)
- template<class \_CharT >  
\_CharT **\_\_gnu\_cxx::\_S\_eos** (\_CharT \*)
- template<class \_CharT >  
bool **\_\_gnu\_cxx::\_S\_is\_basic\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::\_S\_is\_basic\_char\_type** (char \*)
- bool **\_\_gnu\_cxx::\_S\_is\_basic\_char\_type** (wchar\_t \*)
- template<class \_CharT >  
bool **\_\_gnu\_cxx::\_S\_is\_one\_byte\_char\_type** (\_CharT \*)
- bool **\_\_gnu\_cxx::\_S\_is\_one\_byte\_char\_type** (char \*)
- template<class \_CharT, class \_Alloc >  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_x, const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc >  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_x, const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc >  
bool **\_\_gnu\_cxx::operator!=** (const rope< \_CharT, \_Alloc > &\_\_x, const rope< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc >  
bool **\_\_gnu\_cxx::operator!=** (const \_Rope\_char\_ptr\_proxy< \_CharT, \_Alloc > &\_\_x, const \_Rope\_char\_ptr\_proxy< \_CharT, \_Alloc > &\_\_y)
- template<class \_CharT, class \_Alloc >  
\_Rope\_const\_iterator< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_x, std::ptrdiff\_t \_\_n)
- template<class \_CharT, class \_Alloc >  
\_Rope\_const\_iterator< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (std::ptrdiff\_t \_\_n, const \_Rope\_const\_iterator< \_CharT, \_Alloc > &\_\_x)
- template<class \_CharT, class \_Alloc >  
\_Rope\_iterator< \_CharT, \_Alloc > **\_\_gnu\_cxx::operator+** (const \_Rope\_iterator< \_CharT, \_Alloc > &\_\_x, std::ptrdiff\_t \_\_n)



- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator+ (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::operator+ (const rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, const _CharT *__right)`
- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > & __gnu_cxx::operator+= (rope< _CharT, _Alloc > &__left, _CharT __right)`
- `template<class _CharT, class _Alloc >`  
`_Rope_const_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`std::ptrdiff_t __gnu_cxx::operator- (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`_Rope_iterator< _CharT, _Alloc > __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)`
- `template<class _CharT, class _Alloc >`  
`std::ptrdiff_t __gnu_cxx::operator- (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > & __gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`

- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator== (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`bool __gnu_cxx::operator>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (_Rope_char_ref_proxy< _CharT, _Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)`
- `template<class _CharT, class _Alloc >`  
`void __gnu_cxx::swap (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)`

## Variables

- `template<class _CharT, class _Alloc >`  
`rope< _CharT, _Alloc > __gnu_cxx::identity_element (_Rope_Concat_fn< _CharT, _Alloc >)`

## 5.498.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGL STL subset).

## 5.499 ropeimpl.h File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<class _CharT, class _Traits >`  
`void __gnu_cxx::Rope_fill (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)`
- `template<class _CharT >`  
`bool __gnu_cxx::Rope_is_simple (_CharT *)`
- `bool __gnu_cxx::Rope_is_simple (char *)`
- `bool __gnu_cxx::Rope_is_simple (wchar_t *)`
- `template<class _Rope_iterator >`  
`void __gnu_cxx::Rope_rotate (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)`
- `template<class _CharT, class _Traits, class _Alloc >`  
`std::basic_ostream< _CharT, _Traits > &__gnu_cxx::operator<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)`
- `void __gnu_cxx::rotate (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)`

### 5.499.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/rope>`.

## 5.500 rotate\_fn\_imps.hpp File Reference

### 5.500.1 Detailed Description

Contains imps for rotating nodes.

## 5.501 rotate\_fn\_imps.hpp File Reference

### 5.501.1 Detailed Description

Contains imps for rotating nodes.

## 5.502 safe\_base.h File Reference

## Classes

- class `__gnu_debug::Safe_iterator_base`
- class `__gnu_debug::Safe_sequence_base`

## Namespaces

- `__gnu_debug`

## Functions

- [bool \\_\\_gnu\\_debug::\\_\\_check\\_singular\\_aux](#) (const \_Safe\_iterator\_base \*\_\_x)

### 5.502.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.503 [safe\\_container.h](#) File Reference

#### Classes

- [class \\_\\_gnu\\_debug::\\_Safe\\_container<\\_SafeContainer, \\_Alloc, \\_SafeBase, \\_IsCxx11AllocatorAware >](#)

#### Namespaces

- [\\_\\_gnu\\_debug](#)

### 5.503.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.504 [safe\\_iterator.h](#) File Reference

#### Classes

- [struct \\_\\_gnu\\_debug::\\_BeforeBeginHelper<\\_Sequence >](#)
- [class \\_\\_gnu\\_debug::\\_Safe\\_iterator<\\_Iterator, \\_Sequence, \\_Category >](#)
- [struct \\_\\_gnu\\_debug::\\_Sequence\\_traits<\\_Sequence >](#)

#### Namespaces

- [\\_\\_gnu\\_debug](#)

#### Macros

- [#define \\_GLIBCXX\\_DEBUG\\_VERIFY\\_DIST\\_OPERANDS\(\\_Lhs, \\_Rhs\)](#)
- [#define \\_GLIBCXX\\_DEBUG\\_VERIFY\\_EQ\\_OPERANDS\(\\_Lhs, \\_Rhs\)](#)
- [#define \\_GLIBCXX\\_DEBUG\\_VERIFY\\_OPERANDS\(\\_Lhs, \\_Rhs, \\_BadMsgId, \\_DiffMsgId\)](#)
- [#define \\_GLIBCXX\\_DEBUG\\_VERIFY\\_REL\\_OPERANDS\(\\_Lhs, \\_Rhs\)](#)

## Functions

- `template<typename _Iterator, typename _Sequence >`  
`_Iterator \_\_gnu\_debug::\_\_base (const _Safe_iterator< _Iterator, _Sequence, std::random\_access\_iterator\_tag`  
`> &__it)`
- `template<typename _Iterator, typename _Sequence, typename _Category, typename _Size >`  
`bool \_\_gnu\_debug::\_\_can\_advance (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)`
- `template<typename _Iterator, typename _Sequence >`  
`_Iterator \_\_gnu\_debug::\_\_unsafe (const _Safe_iterator< _Iterator, _Sequence > &__it)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_↵`  
`iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence, typename _Category >`  
`bool \_\_gnu\_debug::\_\_valid\_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_↵`  
`_iterator< _Iterator, _Sequence, _Category > &)`

## 5.504.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.505 safe\_iterator.tcc File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)

## Macros

- `#define GLIBCXX\_DEBUG\_SAFE\_ITERATOR\_TCC`

## Functions

- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI std::\_\_copy\_move\_a (const ::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, const ↵`  
`::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > std::\_\_copy\_move\_a (_II, _II, const ::\_\_gnu\_debug::\_\_Safe\_iterator<`  
`_Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::\_\_gnu\_debug::\_\_Safe\_iterator< _OIte, _OSeq, _OCat > std::\_\_copy\_move\_a (const ::\_\_gnu\_debug::\_\_Safe\_iterator<`  
`_IIte, _ISeq, _ICat > &, const ::\_\_gnu\_debug::\_\_Safe\_iterator< _IIte, _ISeq, _ICat > &, const ↵`  
`::\_\_gnu\_debug::\_\_Safe\_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI std::\_\_copy\_move\_backward\_a (const ::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, const ↵`  
`::\_\_gnu\_debug::\_\_Safe\_iterator< _Ite, _Seq, _Cat > &, _OI)`

- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > std::copy_move_backward_a (_II, _II, const ↵`  
`::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::__gnu_debug::__Safe_iterator< _OIte, _OSeq, _OCat > std::copy_move_backward_a (const ↵`  
`::__gnu_debug::__Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::__Safe_iterator< _IIte, _ISeq,`  
`_ICat > &, const ::__gnu_debug::__Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2 >`  
`bool std::equal_aux (const ::__gnu_debug::__Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator<`  
`_II1, _Seq1, _Cat1 > &, _II2)`
- `template<typename _II1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool std::equal_aux (_II1, _II1, const ::__gnu_debug::__Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool std::equal_aux (const ::__gnu_debug::__Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator<`  
`_II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Tp >`  
`void std::fill_a (const ::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::__Safe_iterator<`  
`_Ite, _Seq, _Cat > &, const _Tp &)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp >`  
`::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > std::fill_n_a (const ::__gnu_debug::__Safe_iterator< _Ite,`  
`_Seq, _Cat > & __first, _Size __n, const _Tp & __value, std::input_iterator_tag)`

#### 5.505.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.506 `safe_local_iterator.h` File Reference

#### Classes

- class `__gnu_debug::__Safe_local_iterator< _Iterator, _Sequence >`

#### Namespaces

- `__gnu_debug`

#### Macros

- `#define _GLIBCXX_DEBUG_VERIFY_OPERANDS(_Lhs, _Rhs)`

#### Functions

- `template<typename _Iterator, typename _Sequence >`  
`_Iterator __gnu_debug::unsafe (const _Safe_local_iterator< _Iterator, _Sequence > & __it)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_↵`  
`iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)`
- `template<typename _Iterator, typename _Sequence >`  
`bool __gnu_debug::valid_range (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_↵`  
`_iterator< _Iterator, _Sequence > &)`

## 5.506.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.507 `safe_local_iterator.tcc` File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC`

## 5.507.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.508 `safe_sequence.h` File Reference

## Classes

- class [\\_\\_gnu\\_debug::\\_After\\_nth\\_from<\\_Iterator>](#)
- class [\\_\\_gnu\\_debug::\\_Equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::\\_Not\\_equal\\_to<\\_Type>](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_node\\_sequence<\\_Sequence>](#)
- class [\\_\\_gnu\\_debug::\\_Safe\\_sequence<\\_Sequence>](#)

## Namespaces

- [\\_\\_gnu\\_debug](#)

## 5.508.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

5.509 `safe_sequence.tcc` File Reference

## Namespaces

- [\\_\\_gnu\\_debug](#)

## Macros

- `#define _GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC`

### 5.509.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.510 `safe_unordered_base.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator\\_base](#)
- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container\\_base](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### 5.510.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.511 `safe_unordered_container.h` File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container<\\_Container>](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### 5.511.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.512 `safe_unordered_container.tcc` File Reference

### Namespaces

- [\\_\\_gnu\\_debug](#)



## Macros

- `#define _GLIBCXX_DEBUG_SAFE_UNORDERED_CONTAINER_TCC`

## 5.512.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.513 sample\_probe\_fn.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_probe\\_fn](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.513.1 Detailed Description

Contains a sample probe policy.

## 5.514 sample\_range\_hashing.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_range\\_hashing](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.514.1 Detailed Description

Contains a range hashing policy.

## 5.515 sample\_ranged\_hash\_fn.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_hash\\_fn](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.515.1 Detailed Description

Contains a ranged hash policy.

### 5.516 `sample_ranged_probe_fn.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_ranged\\_probe\\_fn](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.516.1 Detailed Description

Contains a ranged probe policy.

### 5.517 `sample_resize_policy.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_policy](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.517.1 Detailed Description

Contains a sample resize policy for hash tables.

### 5.518 `sample_resize_trigger.hpp` File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::sample\\_resize\\_trigger](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.518.1 Detailed Description

Contains a sample resize trigger policy class.

## 5.519 sample\_size\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_size\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.519.1 Detailed Description

Contains a sample size resize-policy.

## 5.520 sample\_tree\_node\_update.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::sample\\_tree\\_node\\_update< Const\\_Node\\_Iter, Node\\_Iter, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.520.1 Detailed Description

Contains a samle node update functor.

## 5.521 sample\_trie\_access\_traits.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::sample\\_trie\\_access\\_traits](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.521.1 Detailed Description

Contains a sample probe policy.

## 5.522 `sample_trie_node_update.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::sample\\_trie\\_node\\_update](#)< [Node\\_Cltr](#), [Node\\_Itr](#), [\\_ATraits](#), [\\_Alloc](#) >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.522.1 Detailed Description

Contains a samle node update functor.

## 5.523 `sample_update_policy.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::sample\\_update\\_policy](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.523.1 Detailed Description

Contains a sample policy for list update containers.

## 5.524 `scoped_allocator` File Reference

### Classes

- class [std::scoped\\_allocator\\_adaptor](#)< [\\_OuterAlloc](#), [\\_InnerAllocs](#) >

## Namespaces

- [std](#)

## Macros

- `#define _SCOPED_ALLOCATOR`

### 5.524.1 Detailed Description

This is a Standard C++ Library header.

## 5.525 search.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<typename _RAIter, typename _DifferenceTp >  
void \_\_gnu\_parallel::\_\_calc\_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)`
- `template<typename __RAIter1, typename __RAIter2, typename _Pred >  
__RAIter1 \_\_gnu\_parallel::\_\_search\_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, ↔  
__RAIter2 __end2, _Pred __pred)`

### 5.525.1 Detailed Description

Parallel implementation base for `std::search()` and `std::search_n()`. This file is a GNU parallel extension to the Standard C++ Library.

## 5.526 set File Reference

## Macros

- `#define _GLIBCXX_SET`

### 5.526.1 Detailed Description

This is a Standard C++ Library header.

## 5.527 set File Reference

### Classes

- class `std::__debug::multiset<_Key, _Compare, _Allocator >`
- class `std::__debug::set<_Key, _Compare, _Allocator >`

### Namespaces

- `std`
- `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_SET`

#### 5.527.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.528 set File Reference

### Namespaces

- `std`
- `std::experimental`

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_SET`

### Typedefs

- `template<typename _Key, typename _Compare = less<_Key>>`  
using `std::experimental::fundamentals_v2::pmr::multiset` = `std::multiset<_Key, _Compare, polymorphic_allocator<_Key> >`
- `template<typename _Key, typename _Compare = less<_Key>>`  
using `std::experimental::fundamentals_v2::pmr::set` = `std::set<_Key, _Compare, polymorphic_allocator<_Key> >`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`
- `template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)`

## 5.528.1 Detailed Description

This is a TS C++ Library header.

## 5.529 set.h File Reference

## Classes

- class `std::__debug::set< _Key, _Compare, _Allocator >`

## Namespaces

- `std`
- `std::__debug`

## Functions

- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator!= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator< (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator<= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator== (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator> (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`bool std::__debug::operator>= (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)`
- `template<typename _Key, typename _Compare, typename _Allocator >`  
`void std::__debug::swap (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y)`  
`noexcept(/*conditional */)`

### 5.529.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.530 `set_operations.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, typename _OutputIterator >  
_OutputIterator __gnu_parallel::__copy_tail (std::pair< _Iter, _Iter > __b, std::pair< _Iter, _Iter > __e, _↵  
_OutputIterator __r)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >  
_OutputIterator __gnu_parallel::__parallel_set_difference (_Iter __begin1, _Iter __end1, _Iter __begin2, _↵  
_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >  
_OutputIterator __gnu_parallel::__parallel_set_intersection (_Iter __begin1, _Iter __end1, _Iter __begin2, ↵  
_Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Operation >  
_OutputIterator __gnu_parallel::__parallel_set_operation (_Iter __begin1, _Iter __end1, _Iter __begin2, _↵  
_Iter __end2, _OutputIterator __result, _Operation __op)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >  
_OutputIterator __gnu_parallel::__parallel_set_symmetric_difference (_Iter __begin1, _Iter __end1, _Iter ↵  
__begin2, _Iter __end2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Iter, typename _OutputIterator, typename _Compare >  
_OutputIterator __gnu_parallel::__parallel_set_union (_Iter __begin1, _Iter __end1, _Iter __begin2, _Iter ↵  
__end2, _OutputIterator __result, _Compare __comp)`

### 5.530.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

## 5.531 `settings.h` File Reference

### Classes

- `struct \_\_gnu\_parallel::Settings`

### Namespaces

- [\\_\\_gnu\\_parallel](#)



## Macros

- `#define _GLIBCXX_PARALLEL_CONDITION(__c)`

### 5.531.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms.

This file is a GNU parallel extension to the Standard C++ Library.

### 5.531.2 Deciding whether to run an algorithm in parallel.

There are several ways the user can switch on and off the parallel execution of an algorithm, both at compile- and run-time.

Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.

Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.

To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag `gnu_parallel::sequential_tag()` to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.

If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable `__gnu_parallel::_Settings::algorithm_strategy` is checked. It is a tristate variable corresponding to:

- a. `force_sequential`, meaning the sequential algorithm is executed.
- b. `force_parallel`, meaning the parallel algorithm is executed.
- c. `heuristic`

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(`__s`). The threshold can be set by the user, individually for each algorithm. The according variables are called `gnu_parallel::_Settings::[algorithm]_minimal_n`.

For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

### 5.531.3 Macro Definition Documentation

#### 5.531.3.1 \_GLIBCXX\_PARALLEL\_CONDITION

```
#define _GLIBCXX_PARALLEL_CONDITION(
 __c)
```

Determine at compile(?) -time if the parallel variant of an algorithm should be called.

#### Parameters

|                   |                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------|
| <a href="#">↩</a> | A condition that is convertible to bool that is overruled by <code>__gnu_parallel::_Settings::algorithm_strategy</code> . |
| <code>_c</code>   | Usually a decision based on the input size.                                                                               |

Definition at line 94 of file settings.h.

### 5.532 `shared_mutex` File Reference

#### Classes

- class [std::shared\\_lock<\\_Mutex>](#)
- class [std::shared\\_timed\\_mutex](#)

#### Namespaces

- [std](#)

#### Macros

- `#define __cpp_lib_shared_timed_mutex`
- `#define _GLIBCXX_SHARED_MUTEX`

#### 5.532.1 Detailed Description

This is a Standard C++ Library header.

### 5.533 `shared_ptr.h` File Reference

#### Classes

- class [std::enable\\_shared\\_from\\_this<\\_Tp>](#)
- struct [std::hash<shared\\_ptr<\\_Tp>>](#)
- struct [std::owner\\_less<\\_Tp>](#)
- struct [std::owner\\_less<shared\\_ptr<\\_Tp>>](#)
- struct [std::owner\\_less<void>](#)
- struct [std::owner\\_less<weak\\_ptr<\\_Tp>>](#)
- class [std::shared\\_ptr<\\_Tp>](#)
- class [std::weak\\_ptr<\\_Tp>](#)

#### Namespaces

- [std](#)

## Macros

- `#define __cpp_lib_enable_shared_from_this`

## Functions

- `template<typename _Del, typename _Tp, _Lock_policy _Lp>  
_Del * std::get_deleter (const __shared_ptr< _Tp, _Lp > &__p) noexcept`

## 5.533.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.534 shared\_ptr.h File Reference

## Classes

- `struct std::hash< experimental::shared\_ptr< \_Tp > >`
- `struct std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >`
- `struct std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >`

## Namespaces

- [std](#)
- [std::experimental](#)

## Functions

- `template<typename _Tp >  
bool std::experimental::fundamentals_v2::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p,  
shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp >  
bool std::experimental::fundamentals_v2::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp  
> *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __↔  
failure)`
- `template<typename _Tp >  
bool std::experimental::fundamentals_v2::atomic_compare_exchange_weak (shared_ptr< _Tp > *__↔  
p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)`
- `template<typename _Tp >  
bool std::experimental::fundamentals_v2::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp >  
* __p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)`
- `template<typename _Tp >  
void std::experimental::fundamentals_v2::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp >  
__r)`

- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_exchange_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::atomic_is_lock_free (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_load (const shared_ptr< _Tp > *__p)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order __mo)`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)`
- `template<typename _Tp >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::atomic_store_explicit (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::const_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp, typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::dynamic_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Del, typename _Tp >`  
`_Del * std::experimental::fundamentals_v2::get_deleter (const shared_ptr< _Tp > &__p) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator!= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator!= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator< (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator< (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator< (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Ch, typename _Tr, typename _Tp >`  
`std::basic\_ostream< _Ch, _Tr > & std::experimental::fundamentals_v2::operator<< (std::basic\_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator<= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator<= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator<= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator== (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`

- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator== (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator> (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator> (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator> (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp1 , typename _Tp2 >`  
`bool std::experimental::fundamentals_v2::operator>= (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator>= (const shared_ptr< _Tp > &__a, nullptr_t) noexcept`
- `template<typename _Tp >`  
`bool std::experimental::fundamentals_v2::operator>= (nullptr_t, const shared_ptr< _Tp > &__a) noexcept`
- `template<typename _Tp , typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::reinterpret_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp , typename _Tp1 >`  
`shared_ptr< _Tp > std::experimental::fundamentals_v2::static_pointer_cast (const shared_ptr< _Tp1 > &__r) noexcept`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept`
- `template<typename _Tp >`  
`void std::experimental::fundamentals_v2::swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept`

## Variables

- `template<typename _Yp , typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v2::__sp_compatible_v`
- `template<typename _Tp , typename _Yp >`  
`constexpr bool std::experimental::fundamentals_v2::__sp_is_constructible_v`

### 5.534.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/memory>`.

### 5.534.2 Function Documentation

#### 5.534.2.1 `get_deleter()`

```
template<typename _Del , typename _Tp >
_Del* std::experimental::fundamentals_v2::get_deleter (
 const shared_ptr< _Tp > & __p) [inline], [noexcept]
```

C++14 20.8.2.2.10.

Definition at line 492 of file `experimental/bits/shared_ptr.h`.

### 5.535 `shared_ptr_atomic.h` File Reference

#### Namespaces

- [std](#)

#### 5.535.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

### 5.536 `shared_ptr_base.h` File Reference

#### Classes

- struct [std::\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, false >](#)
- struct [std::\\_Sp\\_ebo\\_helper< \\_Nm, \\_Tp, true >](#)
- class [std::bad\\_weak\\_ptr](#)
- class [std::enable\\_shared\\_from\\_this< \\_Tp >](#)
- struct [std::hash< \\_\\_shared\\_ptr< \\_Tp, \\_Lp > >](#)
- struct [std::owner\\_less< \\_Tp >](#)
- class [std::shared\\_ptr< \\_Tp >](#)
- class [std::weak\\_ptr< \\_Tp >](#)

#### Namespaces

- [std](#)

#### Macros

- `#define __cpp_lib_shared_ptr_arrays`

## Functions

- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>  
__shared_ptr< _Tp, _Lp > std::allocate_shared (const _Alloc &__a, _Args &&... __args)`
- `template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>  
__shared_ptr< _Tp, _Lp > std::make_shared (_Args &&... __args)`
- `void std::throw_bad_weak_ptr ()`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator!= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp, typename _Up, _Lock_policy _Lp>  
bool std::operator< (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator< (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator< (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator<= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator<= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator<= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator== (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator== (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator== (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator> (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator> (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`
- `template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>  
bool std::operator>= (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator>= (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
bool std::operator>= (nullptr_t, const __shared_ptr< _Tp, _Lp > &__a) noexcept`

- `template<typename _Tp, typename _Tp1, _Lock_policy _Lp>  
__shared_ptr< _Tp, _Lp > std::static\_pointer\_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
void std::swap (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept`
- `template<typename _Tp, _Lock_policy _Lp>  
void std::swap (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept`

#### 5.536.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

### 5.537 `size_fn_imps.hpp` File Reference

#### 5.537.1 Detailed Description

Contains implementations of `cc_ht_map_`'s entire container size related functions.

### 5.538 `slice_array.h` File Reference

#### Classes

- class [std::slice](#)
- class [std::slice\\_array< \\_Tp >](#)

#### Namespaces

- [std](#)

#### Macros

- `#define _DEFINE_VALARRAY_OPERATOR(_Op, _Name)`

#### 5.538.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.539 `slist` File Reference

#### Classes

- class [\\_\\_gnu\\_cxx::slist< \\_Tp, \\_Alloc >](#)



## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define _SLIST`

## Functions

- `_Slist_node_base * __gnu_cxx::__slist_make_link (_Slist_node_base * __prev_node, _Slist_node_base * __new_node)`
- `_Slist_node_base * __gnu_cxx::__slist_previous (_Slist_node_base * __head, const _Slist_node_base * __node)`
- `const _Slist_node_base * __gnu_cxx::__slist_previous (const _Slist_node_base * __head, const _Slist_node_base * __node)`
- `_Slist_node_base * __gnu_cxx::__slist_reverse (_Slist_node_base * __node)`
- `std::size_t __gnu_cxx::__slist_size (_Slist_node_base * __node)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __before_first, _Slist_node_base * __before_last)`
- `void __gnu_cxx::__slist_splice_after (_Slist_node_base * __pos, _Slist_node_base * __head)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator!= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator== (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
bool __gnu_cxx::operator>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)`
- `template<class _Tp, class _Alloc >  
void __gnu_cxx::swap (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)`

## 5.539.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

## 5.540 sort.h File Reference

## Namespaces

- [\\_\\_gnu\\_parallel](#)

## Functions

- `template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↵`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵`  
`_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵`  
`_exact_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort↵`  
`_sampling_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __↵`  
`parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort↵`  
`_tag __parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag`  
`__parallelism)`
- `template<bool __stable, typename _RAIter, typename _Compare >`  
`void \_\_gnu\_parallel::\_\_parallel\_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __↵`  
`parallelism)`

### 5.540.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

## 5.541 [span](#) File Reference

### Macros

- `#define \_GLIBCXX\_SPAN`

### 5.541.1 Detailed Description

This is a Standard C++ Library header.

## 5.542 [specfun.h](#) File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define __cpp_lib_math_special_functions`
- `#define` `__STDCPP_MATH_SPEC_FUNCS__`

## Functions

- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)`
- `float __gnu_cxx::airy_aif (float __x)`
- `long double __gnu_cxx::airy_ail (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)`
- `float __gnu_cxx::airy_bif (float __x)`
- `long double __gnu_cxx::airy_bil (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)`
- `float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)`
- `long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)`
- `float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)`
- `long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)`
- `template<typename _Tpa, typename _Tpb >`  
`__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)`
- `float std::betaf (float __a, float __b)`
- `long double std::betal (long double __a, long double __b)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)`
- `float std::comp_ellint_1f (float __k)`
- `long double std::comp_ellint_1l (long double __k)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)`
- `float std::comp_ellint_2f (float __k)`
- `long double std::comp_ellint_2l (long double __k)`
- `template<typename _Tp, typename _Tpn >`  
`__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)`
- `float std::comp_ellint_3f (float __k, float __nu)`
- `long double std::comp_ellint_3l (long double __k, long double __nu)`
- `template<typename _Tpa, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)`
- `long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_if (float __nu, float __x)`
- `long double std::cyl_bessel_il (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_jf (float __nu, float __x)`
- `long double std::cyl_bessel_jl (long double __nu, long double __x)`

- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_bessel_kf (float __nu, float __x)`
- `long double std::cyl_bessel_kl (long double __nu, long double __x)`
- `template<typename _Tpnu, typename _Tp >`  
`__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann ( _Tpnu __nu, _Tp __x)`
- `float std::cyl_neumannf (float __nu, float __x)`
- `long double std::cyl_neumannl (long double __nu, long double __x)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_1f (float __k, float __phi)`
- `long double std::ellint_1l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpp >`  
`__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 ( _Tp __k, _Tpp __phi)`
- `float std::ellint_2f (float __k, float __phi)`
- `long double std::ellint_2l (long double __k, long double __phi)`
- `template<typename _Tp, typename _Tpn, typename _Tpp >`  
`__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 ( _Tp __k, _Tpn __nu, _Tpp __phi)`
- `float std::ellint_3f (float __k, float __nu, float __phi)`
- `long double std::ellint_3l (long double __k, long double __nu, long double __phi)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::expint ( _Tp __x)`
- `float std::expintf (float __x)`
- `long double std::expintl (long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)`
- `float std::hermitef (unsigned int __n, float __x)`
- `long double std::hermitel (unsigned int __n, long double __x)`
- `template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp >`  
`__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg ( _Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)`
- `float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)`
- `long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)`
- `float std::laguerref (unsigned int __n, float __x)`
- `long double std::laguerrel (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)`
- `float std::legendref (unsigned int __l, float __x)`
- `long double std::legendrel (unsigned int __l, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta ( _Tp __s)`
- `float std::riemann_zetaf (float __s)`
- `long double std::riemann_zetal (long double __s)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)`
- `float std::sph_besself (unsigned int __n, float __x)`
- `long double std::sph_bessell (unsigned int __n, long double __x)`
- `template<typename _Tp >`  
`__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)`
- `float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)`

- long double [std::sph\\_legendrel](#) (unsigned int \_\_l, unsigned int \_\_m, long double \_\_theta)
- template<typename \_Tp >  
\_\_gnu\_cxx::\_\_promote< \_Tp >::\_\_type [std::sph\\_neumann](#) (unsigned int \_\_n, \_Tp \_\_x)
- float [std::sph\\_neumannf](#) (unsigned int \_\_n, float \_\_x)
- long double [std::sph\\_neumannl](#) (unsigned int \_\_n, long double \_\_x)

#### 5.542.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>`.

### 5.543 splay\_fn\_imps.hpp File Reference

#### 5.543.1 Detailed Description

Contains an implementation class for `splay_tree_`.

### 5.544 splay\_tree\_.hpp File Reference

#### Classes

- class [\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map](#)< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### Macros

- `#define PB_DS_ASSERT_BASE_NODE_CONSISTENT(_Node)`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_S_TREE_BASE`
- `#define PB_DS_S_TREE_BASE_NAME`
- `#define PB_DS_S_TREE_NAME`

#### 5.544.1 Detailed Description

Contains an implementation class for splay trees.

## 5.545 `split_fn_imps.hpp` File Reference

### 5.545.1 Detailed Description

Contains an implementation class for `pat_trie`.

## 5.546 `split_join_fn_imps.hpp` File Reference

### 5.546.1 Detailed Description

Contains an implementation class for a `binary_heap`.

## 5.547 `split_join_fn_imps.hpp` File Reference

### 5.547.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

## 5.548 `split_join_fn_imps.hpp` File Reference

### 5.548.1 Detailed Description

Contains an implementation class for `bin_search_tree_`.

## 5.549 `split_join_fn_imps.hpp` File Reference

### 5.549.1 Detailed Description

Contains an implementation class for `ov_tree_`.

## 5.550 `split_join_fn_imps.hpp` File Reference

### 5.550.1 Detailed Description

Contains an implementation class for a pairing heap.

## 5.551 split\_join\_fn\_imps.hpp File Reference

### 5.551.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.552 split\_join\_fn\_imps.hpp File Reference

### 5.552.1 Detailed Description

Contains an implementation for rc\_binomial\_heap\_.

## 5.553 split\_join\_fn\_imps.hpp File Reference

### 5.553.1 Detailed Description

Contains an implementation class for splay\_tree\_.

## 5.554 split\_join\_fn\_imps.hpp File Reference

### 5.554.1 Detailed Description

Contains an implementation for thin\_heap\_.

## 5.555 sso\_string\_base.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.555.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.556 sstream File Reference

### Classes

- class [std::basic\\_istream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_ostringstream<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringbuf<\\_CharT, \\_Traits, \\_Alloc>](#)
- class [std::basic\\_stringstream<\\_CharT, \\_Traits, \\_Alloc>](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_SSTREAM`

## Functions

- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_istreamstream< _CharT, _Traits, _Allocator > &__x, basic_istreamstream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)`
- `template<class _CharT, class _Traits, class _Allocator >  
void std::swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)`

### 5.556.1 Detailed Description

This is a Standard C++ Library header.

## 5.557 sstream.tcc File Reference

## Namespaces

- [std](#)

## Macros

- `#define _SSTREAM_TCC`

### 5.557.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<sstream>`.



## 5.558 **stack** File Reference

### Macros

- `#define _GLIBCXX_STACK`

#### 5.558.1 Detailed Description

This is a Standard C++ Library header.

## 5.559 **standard\_policies.hpp** File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::default\\_comb\\_hash\\_fn](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_eq\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_hash\\_fn< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_probe\\_fn< Comb\\_Probe\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_resize\\_policy< Comb\\_Hash\\_Fn >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< Key >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_trie\\_access\\_traits< std::basic\\_string< Char, Char\\_Traits, std::allocator< char > > >](#)
- struct [\\_\\_gnu\\_pbds::detail::default\\_update\\_policy](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define __dtrie_alloc`
- `#define __dtrie_string`

### Enumerations

- enum { **default\_store\_hash** }

#### 5.559.1 Detailed Description

Contains standard policies for containers.

#### 5.559.2 Enumeration Type Documentation

### 5.559.2.1 anonymous enum

`anonymous enum`

Enumeration for default behavior of stored hash data.

Definition at line 74 of file `standard_policies.hpp`.

## 5.560 std\_abs.h File Reference

### Namespaces

- [std](#)

### Functions

- long **std::abs** (long \_\_i)
- long long **std::abs** (long long \_\_x)
- constexpr double **std::abs** (double \_\_x)
- constexpr float **std::abs** (float \_\_x)
- constexpr long double **std::abs** (long double \_\_x)

### 5.560.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<cmath>` or `<cstdlib>`.

## 5.561 std\_function.h File Reference

### Classes

- struct [std::\\_\\_is\\_location\\_invariant<\\_Tp>](#)
- class [std::\\_Function\\_base](#)
- class [std::bad\\_function\\_call](#)
- class [std::function<\\_Res\(\\_ArgTypes...\)>](#)

### Namespaces

- [std](#)

### Enumerations

- enum **\_Manager\_operation** { **\_\_get\_type\_info**, **\_\_get\_functor\_ptr**, **\_\_clone\_functor**, **\_\_destroy\_functor** }

## Functions

- template<typename \_Res, typename... \_Args>  
bool [std::operator!=](#) (const function< \_Res(\_Args...)> &\_\_f, nullptr\_t) noexcept
- template<typename \_Res, typename... \_Args>  
bool [std::operator!=](#) (nullptr\_t, const function< \_Res(\_Args...)> &\_\_f) noexcept
- template<typename \_Res, typename... \_Args>  
bool [std::operator==](#) (const function< \_Res(\_Args...)> &\_\_f, nullptr\_t) noexcept
- template<typename \_Res, typename... \_Args>  
bool [std::operator==](#) (nullptr\_t, const function< \_Res(\_Args...)> &\_\_f) noexcept
- template<typename \_Res, typename... \_Args>  
void [std::swap](#) (function< \_Res(\_Args...)> &\_\_x, function< \_Res(\_Args...)> &\_\_y) noexcept

### 5.561.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.562 std\_mutex.h File Reference

## Classes

- struct [std::adopt\\_lock\\_t](#)
- struct [std::defer\\_lock\\_t](#)
- class [std::lock\\_guard](#)< \_Mutex >
- class [std::mutex](#)
- struct [std::try\\_to\\_lock\\_t](#)

## Namespaces

- [std](#)

## Variables

- constexpr adopt\_lock\_t [std::adopt\\_lock](#)
- constexpr defer\_lock\_t [std::defer\\_lock](#)
- constexpr try\_to\_lock\_t [std::try\\_to\\_lock](#)

### 5.562.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 5.563 `stdc++.h` File Reference

### 5.563.1 Detailed Description

This is an implementation file for a precompiled header.

## 5.564 `stdexcept` File Reference

### Classes

- class [std::domain\\_error](#)
- class [std::invalid\\_argument](#)
- class [std::length\\_error](#)
- class [std::logic\\_error](#)
- class [std::out\\_of\\_range](#)
- class [std::overflow\\_error](#)
- class [std::range\\_error](#)
- class [std::runtime\\_error](#)
- class [std::underflow\\_error](#)

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX_STDEXCEPT`

### Typedefs

- typedef `basic_string< char >` **std::\_\_cow\_string**
- typedef `basic_string< char >` **std::\_\_sso\_string**

### 5.564.1 Detailed Description

This is a Standard C++ Library header.

## 5.565 `stdio_filebuf.h` File Reference

### Classes

- class [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.565.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.566 `stdio_sync_filebuf.h` File Reference

## Classes

- class [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## 5.566.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

5.567 `stdlib.h` File Reference

## 5.567.1 Detailed Description

This is a Standard C++ Library header.

5.568 `stdtr1c++.h` File Reference

## 5.568.1 Detailed Description

This is an implementation file for a precompiled header.

5.569 `stl_algo.h` File Reference

## Namespaces

- [std](#)

## Enumerations

- enum { **\_S\_threshold** }
- enum { **\_S\_chunk\_size** }

## Functions

- template<typename **\_ForwardIterator**, typename **\_BinaryPredicate** >  
constexpr **\_ForwardIterator** **std::adjacent\_find** (**\_ForwardIterator** \_\_first, **\_ForwardIterator** \_\_last, **\_BinaryPredicate** \_\_binary\_pred)
- template<typename **\_RandomAccessIterator**, typename **\_Distance**, typename **\_Compare** >  
constexpr void **std::chunk\_insertion\_sort** (**\_RandomAccessIterator** \_\_first, **\_RandomAccessIterator** \_\_last, **\_Distance** \_\_chunk\_size, **\_Compare** \_\_comp)
- template<typename **\_InputIterator**, typename **\_Size**, typename **\_OutputIterator** >  
constexpr **\_OutputIterator** **std::copy\_n** (**\_InputIterator** \_\_first, **\_Size** \_\_n, **\_OutputIterator** \_\_result, input\_iterator\_tag)
- template<typename **\_RandomAccessIterator**, typename **\_Size**, typename **\_OutputIterator** >  
constexpr **\_OutputIterator** **std::copy\_n** (**\_RandomAccessIterator** \_\_first, **\_Size** \_\_n, **\_OutputIterator** \_\_result, random\_access\_iterator\_tag)
- template<typename **\_InputIterator**, typename **\_Size**, typename **\_OutputIterator** >  
constexpr **\_OutputIterator** **std::copy\_n\_a** (**\_InputIterator** \_\_first, **\_Size** \_\_n, **\_OutputIterator** \_\_result)
- template<typename **\_CharT**, typename **\_Size** >  
\_\_enable\_if\_t<\_\_is\_char< **\_CharT** >::value, **\_CharT** \* > **std::copy\_n\_a** (istreambuf\_iterator< **\_CharT**, char\_traits< **\_CharT** >>, **\_Size**, **\_CharT** \*)
- template<typename **\_ForwardIterator**, typename **\_Tp**, typename **\_CompareItTp**, typename **\_CompareTplt** >  
constexpr pair< **\_ForwardIterator**, **\_ForwardIterator** > **std::equal\_range** (**\_ForwardIterator** \_\_first, **\_ForwardIterator** \_\_last, const **\_Tp** & \_\_val, **\_CompareItTp** \_\_comp\_it\_val, **\_CompareTplt** \_\_comp\_val\_it)
- template<typename **\_RandomAccessIterator**, typename **\_Compare** >  
constexpr void **std::final\_insertion\_sort** (**\_RandomAccessIterator** \_\_first, **\_RandomAccessIterator** \_\_last, **\_Compare** \_\_comp)
- template<typename **\_ForwardIterator1**, typename **\_ForwardIterator2**, typename **\_BinaryPredicate** >  
constexpr **\_ForwardIterator1** **std::find\_end** (**\_ForwardIterator1** \_\_first1, **\_ForwardIterator1** \_\_last1, **\_ForwardIterator2** \_\_first2, **\_ForwardIterator2** \_\_last2, forward\_iterator\_tag, forward\_iterator\_tag, **\_BinaryPredicate** \_\_comp)
- template<typename **\_BidirectionalIterator1**, typename **\_BidirectionalIterator2**, typename **\_BinaryPredicate** >  
constexpr **\_BidirectionalIterator1** **std::find\_end** (**\_BidirectionalIterator1** \_\_first1, **\_BidirectionalIterator1** \_\_last1, **\_BidirectionalIterator2** \_\_first2, **\_BidirectionalIterator2** \_\_last2, bidirectional\_iterator\_tag, bidirectional\_iterator\_tag, **\_BinaryPredicate** \_\_comp)
- template<typename **\_InputIterator**, typename **\_Predicate** >  
constexpr **\_InputIterator** **std::find\_if\_not** (**\_InputIterator** \_\_first, **\_InputIterator** \_\_last, **\_Predicate** \_\_pred)
- template<typename **\_InputIterator**, typename **\_Predicate**, typename **\_Distance** >  
constexpr **\_InputIterator** **std::find\_if\_not\_n** (**\_InputIterator** \_\_first, **\_Distance** & \_\_len, **\_Predicate** \_\_pred)
- template<typename **\_EuclideanRingElement** >  
constexpr **\_EuclideanRingElement** **std::gcd** (**\_EuclideanRingElement** \_\_m, **\_EuclideanRingElement** \_\_n)
- template<typename **\_IntType**, typename **\_UniformRandomBitGenerator** >  
pair< **\_IntType**, **\_IntType** > **std::gen\_two\_uniform\_ints** (**\_IntType** \_\_b0, **\_IntType** \_\_b1, **\_UniformRandomBitGenerator** && \_\_g)
- template<typename **\_RandomAccessIterator**, typename **\_Compare** >  
constexpr void **std::heap\_select** (**\_RandomAccessIterator** \_\_first, **\_RandomAccessIterator** \_\_middle, **\_RandomAccessIterator** \_\_last, **\_Compare** \_\_comp)
- template<typename **\_InputIterator1**, typename **\_InputIterator2**, typename **\_Compare** >  
constexpr bool **std::includes** (**\_InputIterator1** \_\_first1, **\_InputIterator1** \_\_last1, **\_InputIterator2** \_\_first2, **\_InputIterator2** \_\_last2, **\_Compare** \_\_comp)

- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::insertion_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`constexpr void std::introsort (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Size, typename _Compare >`  
`constexpr void std::introsort_loop (_RandomAccessIterator __first, _RandomAccessIterator __last, _Size __depth_limit, _Compare __comp)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare >`  
`void std::merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare >`  
`void std::merge_sort_loop (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Compare >`  
`void std::merge_sort_with_buffer (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Distance, typename _Compare >`  
`void std::merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr pair<_ForwardIterator, _ForwardIterator> std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Iterator, typename _Compare >`  
`constexpr void std::move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Compare >`  
`_OutputIterator std::move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`void std::move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`

- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare >`  
`void std::__move_merge_adaptive_backward ( _BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, ↵`  
`_BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool std::__next_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare`  
`__comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::__partial_sort ( _RandomAccessIterator __first, _RandomAccessIterator __middle, ↵`  
`_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator std::__partial_sort_copy ( _InputIterator __first, _InputIterator __last, ↵`  
`_RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator std::__partition ( _ForwardIterator __first, _ForwardIterator __last, _Predicate __pred,`  
`forward_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Predicate >`  
`constexpr _BidirectionalIterator std::__partition ( _BidirectionalIterator __first, _BidirectionalIterator __last, ↵`  
`_Predicate __pred, bidirectional_iterator_tag)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool std::__prev_permutation ( _BidirectionalIterator __first, _BidirectionalIterator __last, _Compare`  
`__comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`constexpr _OutputIterator std::__remove_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator std::__remove_if ( _ForwardIterator __first, _ForwardIterator __last, _Predicate ↵`  
`__pred)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`constexpr _OutputIterator std::__replace_copy_if ( _InputIterator __first, _InputIterator __last, _OutputIterator`  
`__result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`constexpr void std::__reverse ( _BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator ↵`  
`__tag)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::__reverse ( _RandomAccessIterator __first, _RandomAccessIterator __last, random_access ↵`  
`__iterator_tag)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::__V2::__rotate ( _ForwardIterator __first, _ForwardIterator __middle, _Forward ↵`  
`__iterator __last, forward_iterator_tag)`
- `template<typename _BidirectionalIterator >`  
`constexpr _BidirectionalIterator std::__V2::__rotate ( _BidirectionalIterator __first, _BidirectionalIterator __middle,`  
`_BidirectionalIterator __last, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`constexpr _RandomAccessIterator std::__V2::__rotate ( _RandomAccessIterator __first, _RandomAccessIterator`  
`__middle, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance >`  
`_BidirectionalIterator1 std::__rotate_adaptive ( _BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, ↵`  
`_BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance`  
`__buffer_size)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator >`  
`_RandomAccessIterator std::__sample ( _InputIterator __first, _InputIterator __last, input_iterator_tag, ↵`  
`_RandomAccessIterator __out, random_access_iterator_tag, _Size __n, _UniformRandomBitGenerator &&__g)`



- `template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBitGenerator>`  
`_OutputIterator std::\_\_sample (_ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _OutputIterator __out, _Cat, _Size __n, _UniformRandomBitGenerator && __g)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>`  
`constexpr _ForwardIterator1 std::\_\_search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::\_\_search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)`
- `template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>`  
`constexpr _ForwardIterator std::\_\_search\_n\_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward\_iterator\_tag)`
- `template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>`  
`constexpr _RandomAccessIter std::\_\_search\_n\_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random\_access\_iterator\_tag)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>`  
`constexpr _OutputIterator std::\_\_set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::\_\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate>`  
`_ForwardIterator std::\_\_stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>`  
`_ForwardIterator std::\_\_stable\_partition\_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`void std::\_\_stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>`  
`void std::\_\_stable\_sort\_adaptive (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::\_\_unguarded\_insertion\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr void std::\_\_unguarded\_linear\_insert (_RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::\_\_unguarded\_partition (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __pivot, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Compare>`  
`constexpr _RandomAccessIterator std::\_\_unguarded\_partition\_pivot (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _BinaryPredicate>`  
`constexpr _ForwardIterator std::\_\_unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)`

- `template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator std::unique\_copy (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator`  
`__result, _BinaryPredicate __binary_pred, forward_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __↵`  
`result, _BinaryPredicate __binary_pred, input_iterator_tag, output_iterator_tag)`
- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __↵`  
`__result, _BinaryPredicate __binary_pred, input_iterator_tag, forward_iterator_tag)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp`  
`&__val, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator std::adjacent\_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate`  
`__binary_pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool std::all\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool std::any\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr bool std::binary\_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare`  
`__comp)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`constexpr _OutputIterator std::copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, __↵`  
`Predicate __pred)`
- `template<typename _InputIterator, typename _Size, typename _OutputIterator >`  
`constexpr _OutputIterator std::copy\_n (_InputIterator __first, _Size __n, _OutputIterator __result)`
- `template<typename _InputIterator, typename _Tp >`  
`constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last,`  
`const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr iterator_traits< _InputIterator >::difference_type std::count\_if (_InputIterator __first, _InputIterator __↵`  
`__last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _Forward↵`  
`Iterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::equal\_range (_ForwardIterator __first, _Forward↵`  
`Iterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _InputIterator, typename _Tp >`  
`constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↵`  
`Iterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr _ForwardIterator1 std::find\_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↵`  
`Iterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _ForwardIterator >`  
`constexpr _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2,`  
`_ForwardIterator __last2)`

- `template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _InputIterator std::find\_first\_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2,`  
`_ForwardIterator __last2, _BinaryPredicate __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr _InputIterator std::find\_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr _InputIterator std::find\_if\_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _Function >`  
`constexpr _Function std::for\_each (_InputIterator __first, _InputIterator __last, _Function __f)`
- `template<typename _ForwardIterator, typename _Generator >`  
`constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)`
- `template<typename _OutputIterator, typename _Size, typename _Generator >`  
`constexpr _OutputIterator std::generate\_n (_OutputIterator __first, _Size __n, _Generator __gen)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, Input↵`  
`Iterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Compare >`  
`constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, Input↵`  
`Iterator2 __last2, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵`  
`last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`void std::inplace\_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↵`  
`last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool std::is\_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵`  
`__first2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵`  
`__first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool std::is\_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵`  
`__first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)`
- `template<typename _ForwardIterator >`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr bool std::is\_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::is\_sorted\_until (_ForwardIterator __first, _ForwardIterator __last, _Compare ↵`  
`__comp)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator std::lower\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,`  
`_Compare __comp)`
- `template<typename _Tp >`  
`constexpr _Tp std::max (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp std::max (initializer_list< _Tp >, _Compare)`

- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::max\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr _Tp std::min (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr _Tp std::min (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr _ForwardIterator std::min\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >)`
- `template<typename _Tp, typename _Compare >`  
`constexpr pair< _Tp, _Tp > std::minmax (initializer_list< _Tp >, _Compare)`
- `template<typename _ForwardIterator >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _Compare >`  
`constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax\_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)`
- `template<typename _BidirectionalIterator >`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool std::next\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr bool std::none\_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::nth\_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::partial\_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _InputIterator, typename _RandomAccessIterator >`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)`
- `template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator std::partial\_sort\_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate >`  
`constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator std::partition\_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _BidirectionalIterator >`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _Compare >`  
`constexpr bool std::prev\_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _RandomNumberGenerator >`  
`void std::random\_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`constexpr _OutputIterator std::remove\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate >`  
`constexpr _OutputIterator std::remove\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`constexpr _ForwardIterator std::remove\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Tp >`  
`constexpr _OutputIterator std::replace\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)`
- `template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp >`  
`constexpr _OutputIterator std::replace\_copy\_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _ForwardIterator, typename _Predicate, typename _Tp >`  
`constexpr void std::replace\_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)`
- `template<typename _BidirectionalIterator >`  
`constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)`
- `template<typename _BidirectionalIterator, typename _OutputIterator >`  
`constexpr _OutputIterator std::reverse\_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::V2::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)`

- `template<typename _ForwardIterator, typename _OutputIterator >`  
`constexpr _OutputIterator std::rotate\_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp >`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)`
- `template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate >`  
`constexpr _ForwardIterator std::search\_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::set\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::set\_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::set\_symmetric\_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator >`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare >`  
`constexpr _OutputIterator std::set\_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator >`  
`void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _ForwardIterator, typename _Predicate >`  
`_ForwardIterator std::stable\_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)`
- `template<typename _RandomAccessIterator >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`void std::stable\_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`

- `template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation >`  
`constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↵`  
`_UnaryOperation __unary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation >`  
`constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,`  
`_OutputIterator __result, _BinaryOperation __binary_op)`
- `template<typename _ForwardIterator >`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _ForwardIterator, typename _BinaryPredicate >`  
`constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↵`  
`binary_pred)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate >`  
`constexpr _OutputIterator std::unique\_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,`  
`_BinaryPredicate __binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`val)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator std::upper\_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↵`  
`val, _Compare __comp)`

### 5.569.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

### 5.569.2 Function Documentation

#### 5.569.2.1 `__rotate()` [1/3]

```
template<typename _ForwardIterator >
constexpr _ForwardIterator std::_V2::__rotate (
 _ForwardIterator __first,
 _ForwardIterator __middle,
 _ForwardIterator __last,
 forward_iterator_tag)
```

This is a helper function for the rotate algorithm.

Definition at line 1242 of file `stl_algo.h`.

References `std::_V2::__rotate()`, and `std::iter_swap()`.

Referenced by `std::_V2::__rotate()`.

### 5.569.2.2 `__rotate()` [2/3]

```
template<typename _BidirectionalIterator >
constexpr _BidirectionalIterator std::_V2::__rotate (
 _BidirectionalIterator __first,
 _BidirectionalIterator __middle,
 _BidirectionalIterator __last,
 bidirectional_iterator_tag)
```

This is a helper function for the rotate algorithm.

Definition at line 1284 of file `stl_algo.h`.

References `std::_V2::__rotate()`.

### 5.569.2.3 `__rotate()` [3/3]

```
template<typename _RandomAccessIterator >
constexpr _RandomAccessIterator std::_V2::__rotate (
 _RandomAccessIterator __first,
 _RandomAccessIterator __middle,
 _RandomAccessIterator __last,
 random_access_iterator_tag)
```

This is a helper function for the rotate algorithm.

Definition at line 1323 of file `stl_algo.h`.

References `std::_V2::__rotate()`.

## 5.570 `stl_algobase.h` File Reference

### Classes

- struct `std::_Deque_iterator<_Tp, _Ref, _Ptr>`
- struct `std::char_traits<_CharT>`
- class `std::istreambuf_iterator<_CharT, _Traits>`
- class `std::ostreambuf_iterator<_CharT, _Traits>`

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_robust_nonmodifying_seq_ops`
- `#define _GLIBCXX_MOVE3(_Tp, _Up, _Vp)`
- `#define _GLIBCXX_MOVE_BACKWARD3(_Tp, _Up, _Vp)`



## Functions

- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI std::copy_move_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI std::copy_move_a (const ::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const ↵`  
`::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`_gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > std::copy_move_a (_II, _II, const ::gnu_debug::Safe_iterator<`  
`_Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > std::copy_move_a (const ::gnu_debug::Safe_iterator<`  
`_IIte, _ISeq, _ICat > &, const ::gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const ↵`  
`::gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI std::copy_move_a1 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std::copy_move_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp, _Ref, _Ptr >`  
`__last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std::copy_move_a1 (::Deque_iterator< _ITp, _IRef, _IPtr >`  
`__first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp * > __result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`_gnu_cxx::enable_if< __is_random_access_iter< _II >::value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::type std::copy_move_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * > __result)`
- `template<bool _IsMove, typename _CharT >`  
`_gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT >`  
`> >::type std::copy_move_a2 (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> >)`
- `template<bool _IsMove, typename _CharT >`  
`_gnu_cxx::enable_if< __is_char< _CharT >::value, ostreambuf_iterator< _CharT, char_traits< _CharT`  
`> > >::type std::copy_move_a2 (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_↵`  
`traits< _CharT > >)`
- `template<bool _IsMove, typename _CharT >`  
`_gnu_cxx::enable_if< __is_char< _CharT >::value, _CharT * >::type std::copy_move_a2`  
`(istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT`  
`> >, _CharT *)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI std::copy_move_a2 (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _II, typename _OI >`  
`constexpr _OI std::copy_move_backward_a (_II __first, _II __last, _OI __result)`
- `template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI >`  
`_OI std::copy_move_backward_a (const ::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, const ↵`  
`::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &, _OI)`
- `template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat >`  
`_gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > std::copy_move_backward_a (_II, _II, const ↵`  
`::gnu_debug::Safe_iterator< _Ite, _Seq, _Cat > &)`
- `template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat >`  
`::gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > std::copy_move_backward_a (const ↵`  
`::gnu_debug::Safe_iterator< _IIte, _ISeq, _ICat > &, const ::gnu_debug::Safe_iterator< _IIte, _ISeq,`  
`_ICat > &, const ::gnu_debug::Safe_iterator< _OIte, _OSeq, _OCat > &)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`constexpr _BI2 std::copy_move_backward_a1 (_BI1 __first, _BI1 __last, _BI2 __result)`

- `template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI >`  
`_OI std::__copy_move_backward_a1 (::Deque_iterator< _Tp, _Ref, _Ptr > __first, ::Deque_iterator< _Tp,`  
`_Ref, _Ptr > __last, _OI __result)`
- `template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp >`  
`::Deque_iterator< _OTp, _OTp &, _OTp * > std::__copy_move_backward_a1 (::Deque_iterator< _ITp, _IRef,`  
`_IPtr > __first, ::Deque_iterator< _ITp, _IRef, _IPtr > __last, ::Deque_iterator< _OTp, _OTp &, _OTp *`  
`> __result)`
- `template<bool _IsMove, typename _II, typename _Tp >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`>::__type std::__copy_move_backward_a1 (_II __first, _II __last, ::Deque_iterator< _Tp, _Tp &, _Tp * >`  
`__result)`
- `template<bool _IsMove, typename _BI1, typename _BI2 >`  
`constexpr _BI2 std::__copy_move_backward_a2 (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr iterator_traits< _InputIterator >::difference_type std::__count_if (_InputIterator __first, _InputIterator`  
`__last, _Predicate __pred)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _BinaryPredicate >`  
`constexpr bool std::__equal4 (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary,`  
`__pred)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::__equal_aux (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2 >`  
`bool std::__equal_aux (const ::__gnu_debug::__Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator<`  
`_II1, _Seq1, _Cat1 > &, _II2)`
- `template<typename _II1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool std::__equal_aux (_II1, _II1, const ::__gnu_debug::__Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2 >`  
`bool std::__equal_aux (const ::__gnu_debug::__Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator<`  
`_II1, _Seq1, _Cat1 > &, const ::__gnu_debug::__Safe_iterator< _II2, _Seq2, _Cat2 > &)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::__equal_aux1 (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _Tp, typename _Ref, typename _Ptr, typename _II >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1 (::Deque_iterator< _Tp, _Ref, _Ptr >`  
`__first1, ::Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)`
- `template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2 >`  
`bool std::__equal_aux1 (::Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::Deque_iterator< _Tp1, _Ref1,`  
`_Ptr1 > __last1, ::Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)`
- `template<typename _II, typename _Tp, typename _Ref, typename _Ptr >`  
`__gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type std::__equal_aux1 (_II __first1, _II __last1,`  
`::Deque_iterator< _Tp, _Ref, _Ptr > __first2)`
- `template<typename _Flte, typename _Tp >`  
`constexpr void std::__fill_a (_Flte __first, _Flte __last, const _Tp & __value)`
- `template<typename _Lte, typename _Seq, typename _Cat, typename _Tp >`  
`void std::__fill_a (const ::__gnu_debug::__Safe_iterator< _Lte, _Seq, _Cat > &, const ::__gnu_debug::__Safe_iterator<`  
`_Lte, _Seq, _Cat > &, const _Tp &)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< !__is_scalar< _Tp >::__value, void >::__type std::__fill_a1 (_ForwardIterator __first,`  
`_ForwardIterator __last, const _Tp & __value)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, void >::__type std::__fill_a1 (_ForwardIterator __first,`  
`_ForwardIterator __last, const _Tp & __value)`

- `template<typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_byte< _Tp >::__value, void >::__type std::__fill_a1 (_Tp *__first, _Tp *__last, const _Tp &__c)`
- `template<typename _Ite, typename _Cont, typename _Tp >`  
`constexpr void std::__fill_a1 (::__gnu_cxx::__normal_iterator< _Ite, _Cont > __first, ::__gnu_cxx::__normal_iterator< _Ite, _Cont > __last, const _Tp &__value)`
- `template<typename _Tp, typename _VTp >`  
`void std::__fill_a1 (const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)`
- `template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp >`  
`::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > std::__fill_n_a (const ::__gnu_debug::__Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr _OutputIterator std::__fill_n_a (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if<! __is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _OutputIterator, typename _Size, typename _Tp >`  
`constexpr __gnu_cxx::__enable_if< __is_scalar< _Tp >::__value, _OutputIterator >::__type std::__fill_n_a1 (_OutputIterator __first, _Size __n, const _Tp &__value)`
- `template<typename _InputIterator, typename _Predicate >`  
`constexpr _InputIterator std::__find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred, input_iterator_tag)`
- `template<typename _RandomAccessIterator, typename _Predicate >`  
`constexpr _RandomAccessIterator std::__find_if (_RandomAccessIterator __first, _RandomAccessIterator __last, _Predicate __pred, random_access_iterator_tag)`
- `template<typename _Iterator, typename _Predicate >`  
`constexpr _Iterator std::__find_if (_Iterator __first, _Iterator __last, _Predicate __pred)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate >`  
`constexpr bool std::__is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::__lexicographical_compare_aux (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`constexpr bool std::__lexicographical_compare_impl (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, __Compare __comp)`
- `constexpr int std::__lg (int __n)`
- `constexpr unsigned std::__lg (unsigned __n)`
- `constexpr long std::__lg (long __n)`
- `constexpr unsigned long std::__lg (unsigned long __n)`
- `constexpr long long std::__lg (long long __n)`
- `constexpr unsigned long long std::__lg (unsigned long long __n)`
- `template<typename _ForwardIterator, typename _Tp, typename _Compare >`  
`constexpr _ForwardIterator std::__lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)`
- `template<typename _Tp, typename _Up >`  
`constexpr int std::__memcmp (const _Tp *__first1, const _Up *__first2, size_t __num)`

- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 ↵`  
`__last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 ↵`  
`__last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _Iterator >`  
`constexpr _Iterator std::niter_base (_Iterator __it) noexcept(/*conditional */)`
- `template<typename _From, typename _To >`  
`constexpr _From std::niter_wrap (_From __from, _To __res)`
- `template<typename _Iterator >`  
`constexpr _Iterator std::niter_wrap (const _Iterator &, _Iterator __res)`
- `constexpr int std::size_to_integer (int __n)`
- `constexpr unsigned std::size_to_integer (unsigned __n)`
- `constexpr long std::size_to_integer (long __n)`
- `constexpr unsigned long std::size_to_integer (unsigned long __n)`
- `constexpr long long std::size_to_integer (long long __n)`
- `constexpr unsigned long long std::size_to_integer (unsigned long long __n)`
- `constexpr long long std::size_to_integer (float __n)`
- `constexpr long long std::size_to_integer (double __n)`
- `constexpr long long std::size_to_integer (long double __n)`
- `template<typename _II, typename _OI >`  
`constexpr _OI std::copy (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`constexpr bool std::equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _Iiter1, typename _Iiter2, typename _BinaryPredicate >`  
`constexpr bool std::equal (_Iiter1 __first1, _Iiter1 __last1, _Iiter2 __first2, _Iiter2 __last2, _BinaryPredicate ↵`  
`binary_pred)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)`
- `template<typename _OI, typename _Size, typename _Tp >`  
`constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↵`  
`__first2)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)`
- `template<typename _II1, typename _II2 >`  
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)`
- `template<typename _II1, typename _II2, typename _Compare >`  
`constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare`  
`__comp)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)`
- `template<typename _Tp >`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)`

- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _Tp >`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)`
- `template<typename _Tp, typename _Compare >`  
`constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)`
- `template<typename _InputIterator1, typename _InputIterator2 >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate >`  
`constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)`
- `template<typename _II, typename _OI >`  
`constexpr _OI std::move (_II __first, _II __last, _OI __result)`
- `template<typename _BI1, typename _BI2 >`  
`constexpr _BI2 std::move\_backward (_BI1 __first, _BI1 __last, _BI2 __result)`
- `template<typename _ForwardIterator1, typename _ForwardIterator2 >`  
`constexpr _ForwardIterator2 std::swap\_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)`

#### 5.570.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<algorithm>`.

## 5.571 `std::bvector.h` File Reference

### Classes

- struct [std::hash<::vector< bool, \\_Alloc > >](#)
- class [std::vector< bool, \\_Alloc >](#)

### Namespaces

- [std](#)

### Typedefs

- typedef unsigned long [std::\\_Bit\\_type](#)

## Enumerations

- `enum { _S_word_bit }`

## Functions

- `void std::fill_bvector (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x)`
- `void std::fill (_Bit_iterator __first, _Bit_iterator __last, const bool &__x)`
- `void std::swap (_Bit_reference __x, _Bit_reference __y) noexcept`
- `void std::swap (_Bit_reference __x, bool &__y) noexcept`
- `void std::swap (bool &__x, _Bit_reference __y) noexcept`

### 5.571.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 5.572 `stl_construct.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _Tp, typename... _Args>  
void std::\_Construct (_Tp *__p, _Args &&... __args)`
- `template<typename _T1 >  
void std::_Construct_novalue (_T1 *__p)`
- `template<typename _ForwardIterator >  
constexpr void std::\_Destroy (_ForwardIterator __first, _ForwardIterator __last)`
- `template<typename _Tp >  
constexpr void std::\_Destroy (_Tp *__pointer)`
- `template<typename _ForwardIterator, typename _Size >  
constexpr _ForwardIterator std::\_Destroy\_n (_ForwardIterator __first, _Size __count)`

### 5.572.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.573 std\_deque.h File Reference

## Classes

- class [std::\\_Deque\\_base<\\_Tp, \\_Alloc>](#)
- struct [std::\\_Deque\\_iterator<\\_Tp, \\_Ref, \\_Ptr>](#)
- class [std::deque<\\_Tp, \\_Alloc>](#)

## Namespaces

- [std](#)

## Macros

- [#define \\_GLIBCXX\\_DEQUE\\_BUF\\_SIZE](#)

## Functions

- constexpr size\_t [std::\\_\\_deque\\_buf\\_size](#) (size\_t \_\_size)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator!=](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator<](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator<=](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator==](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator>](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
bool [std::operator>=](#) (const deque<\_Tp, \_Alloc> &\_\_x, const deque<\_Tp, \_Alloc> &\_\_y)
- template<typename \_Tp, typename \_Alloc>  
void [std::swap](#) (deque<\_Tp, \_Alloc> &\_\_x, deque<\_Tp, \_Alloc> &\_\_y) noexcept(*/\*conditional \*/*)

## 5.573.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<deque>`.

## 5.573.2 Macro Definition Documentation

## 5.573.2.1 \_GLIBCXX\_DEQUE\_BUF\_SIZE

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

#### Parameters

|                     |                         |
|---------------------|-------------------------|
| <code>__size</code> | The size of an element. |
|---------------------|-------------------------|

#### Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch `_GLIBCXX_DEQUE_BUF_SIZE` only if you know what you are doing, however: changing it breaks the binary compatibility!!

Definition at line 92 of file `stl_deque.h`.

### 5.574 `stl_function.h` File Reference

#### Classes

- struct `std::binary_function<_Arg1, _Arg2, _Result >`
- class `std::binary_negate<_Predicate >`
- class `std::const_mem_fun1_ref_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun1_t<_Ret, _Tp, _Arg >`
- class `std::const_mem_fun_ref_t<_Ret, _Tp >`
- class `std::const_mem_fun_t<_Ret, _Tp >`
- struct `std::divides<_Tp >`
- struct `std::divides<_Tp >`
- struct `std::divides<void >`
- struct `std::equal_to<_Tp >`
- struct `std::equal_to<_Tp >`
- struct `std::equal_to<void >`
- struct `std::greater<_Tp >`
- struct `std::greater<_Tp >`
- struct `std::greater<void >`
- struct `std::greater_equal<_Tp >`
- struct `std::greater_equal<_Tp >`
- struct `std::greater_equal<void >`
- struct `std::less<_Tp >`
- struct `std::less<_Tp >`
- struct `std::less<void >`
- struct `std::less_equal<_Tp >`
- struct `std::less_equal<_Tp >`
- struct `std::less_equal<void >`
- struct `std::logical_and<_Tp >`
- struct `std::logical_and<_Tp >`
- struct `std::logical_and<void >`
- struct `std::logical_not<_Tp >`
- struct `std::logical_not<_Tp >`



- struct `std::logical_not< void >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< _Tp >`
- struct `std::logical_or< void >`
- class `std::mem_fun1_ref_t< _Ret, _Tp, _Arg >`
- class `std::mem_fun1_t< _Ret, _Tp, _Arg >`
- class `std::mem_fun_ref_t< _Ret, _Tp >`
- class `std::mem_fun_t< _Ret, _Tp >`
- struct `std::minus< _Tp >`
- struct `std::minus< _Tp >`
- struct `std::minus< void >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< _Tp >`
- struct `std::modulus< void >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< _Tp >`
- struct `std::multiplies< void >`
- struct `std::negate< _Tp >`
- struct `std::negate< _Tp >`
- struct `std::negate< void >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< _Tp >`
- struct `std::not_equal_to< void >`
- struct `std::plus< _Tp >`
- struct `std::plus< _Tp >`
- class `std::pointer_to_binary_function< _Arg1, _Arg2, _Result >`
- class `std::pointer_to_unary_function< _Arg, _Result >`
- struct `std::unary_function< _Arg, _Result >`
- class `std::unary_negate< _Predicate >`

## Namespaces

- `std`

## Macros

- `#define __cpp_lib_transparent_operators`

## Functions

- `template<typename _Ret, typename _Tp >`  
`mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp >`  
`const_mem_fun_t< _Ret, _Tp > std::mem_fun (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg >`  
`const_mem_fun1_t< _Ret, _Tp, _Arg > std::mem_fun (_Ret(_Tp::*__f)(_Arg) const)`

- `template<typename _Ret, typename _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)())`
- `template<typename _Ret, typename _Tp > std::mem_fun_ref (_Ret(_Tp::*__f)() const)`
- `template<typename _Ret, typename _Tp, typename _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg))`
- `template<typename _Ret, typename _Tp, typename _Arg > std::mem_fun_ref (_Ret(_Tp::*__f)(_Arg) const)`
- `template<typename _Predicate > std::not1 (const _Predicate &__pred)`
- `template<typename _Predicate > std::not2 (const _Predicate &__pred)`
- `template<typename _Arg, typename _Result > std::ptr_fun (_Result(*__x)(_Arg))`
- `template<typename _Arg1, typename _Arg2, typename _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))`

#### 5.574.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<functional>`.

## 5.575 `std_heap.h` File Reference

### Namespaces

- [std](#)

### Functions

- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare > std::__adjust_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance > std::__is_heap (_RandomAccessIterator __first, _Distance __n)`
- `template<typename _RandomAccessIterator, typename _Compare, typename _Distance > std::__is_heap (_RandomAccessIterator __first, _Compare __comp, _Distance __n)`
- `template<typename _RandomAccessIterator > std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare > std::__is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Compare > std::__is_heap_until (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare > std::__make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`

- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::__pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare >`  
`constexpr void std::__push_heap (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::__sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`
- `template<typename _RandomAccessIterator >`  
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)`
- `template<typename _RandomAccessIterator, typename _Compare >`  
`constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)`

### 5.575.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

## 5.576 `std_iterator.h` File Reference

### Classes

- class `std::back_insert_iterator< _Container >`
- class `std::front_insert_iterator< _Container >`
- class `std::insert_iterator< _Container >`
- class `std::move_iterator< _Iterator >`
- class `std::reverse_iterator< _Iterator >`

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Macros

- `#define __cpp_lib_make_reverse_iterator`
- `#define GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR(_Iter)`
- `#define GLIBCXX_MAKE_MOVE_ITERATOR(_Iter)`

## Functions

- `template<typename _Iterator, typename _ReturnType = typename conditional<__move_if_noexcept_cond<typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>::type>`  
`constexpr _ReturnType std::make_move_if_noexcept_iterator (_Iterator __i)`
- `template<typename _Tp, typename _ReturnType = typename conditional<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>::type>`  
`constexpr _ReturnType std::make_move_if_noexcept_iterator (_Tp *__i)`
- `template<typename _Iterator >`  
`constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`constexpr auto std::miter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__miter_base(__it.base())))`
- `template<typename _Iterator >`  
`auto std::miter_base (move_iterator< _Iterator > __it) -> decltype(__miter_base(__it.base()))`
- `template<typename _Iterator >`  
`constexpr auto std::niter_base (reverse_iterator< _Iterator > __it) -> decltype(__make_reverse_iterator(__niter_base(__it.base())))`
- `template<typename _Iterator, typename _Container >`  
`constexpr _Iterator std::niter_base (__gnu_cxx::__normal_iterator< _Iterator, _Container > __it) noexcept(*conditional *)`
- `template<typename _Iterator >`  
`auto std::niter_base (move_iterator< _Iterator > __it) -> decltype(make_move_iterator(__niter_base(__it.base())))`
- `template<typename _Container >`  
`constexpr back_insert_iterator< _Container > std::back_inserter (_Container &__x)`
- `template<typename _Container >`  
`constexpr front_insert_iterator< _Container > std::front_inserter (_Container &__x)`
- `template<typename _Container >`  
`insert_iterator< _Container > std::inserter (_Container &__x, typename _Container::iterator __i)`
- `template<typename _Iterator >`  
`constexpr move_iterator< _Iterator > std::make_move_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)`
- `template<typename _Iterator >`  
`constexpr bool std::operator!= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator!= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`

- template<typename \_IteratorL, typename \_IteratorR, typename \_Container >  
constexpr bool **gnu\_cxx::operator!=** (const \_\_normal\_iterator< \_IteratorL, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_IteratorR, \_Container > &\_\_rhs) noexcept
- template<typename \_Iterator, typename \_Container >  
constexpr bool **gnu\_cxx::operator!=** (const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_rhs) noexcept
- template<typename \_IteratorL, typename \_IteratorR >  
constexpr bool **std::operator!=** (const move\_iterator< \_IteratorL > &\_\_x, const move\_iterator< \_IteratorR > &\_\_y)
- template<typename \_Iterator >  
constexpr bool **std::operator!=** (const move\_iterator< \_Iterator > &\_\_x, const move\_iterator< \_Iterator > &\_\_y)
- template<typename \_Iterator >  
constexpr reverse\_iterator< \_Iterator > **std::operator+** (typename reverse\_iterator< \_Iterator >::difference\_type \_\_n, const reverse\_iterator< \_Iterator > &\_\_x)
- template<typename \_Iterator, typename \_Container >  
constexpr \_\_normal\_iterator< \_Iterator, \_Container > **gnu\_cxx::operator+** (typename \_\_normal\_iterator< \_Iterator, \_Container >::difference\_type \_\_n, const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_i) noexcept
- template<typename \_Iterator >  
constexpr move\_iterator< \_Iterator > **std::operator+** (typename move\_iterator< \_Iterator >::difference\_type \_\_n, const move\_iterator< \_Iterator > &\_\_x)
- template<typename \_IteratorL, typename \_IteratorR >  
constexpr auto **std::operator-** (const reverse\_iterator< \_IteratorL > &\_\_x, const reverse\_iterator< \_IteratorR > &\_\_y) -> decltype(\_\_y.base() - \_\_x.base())
- template<typename \_IteratorL, typename \_IteratorR, typename \_Container >  
constexpr auto **gnu\_cxx::operator-** (const \_\_normal\_iterator< \_IteratorL, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_IteratorR, \_Container > &\_\_rhs) noexcept -> decltype(\_\_lhs.base() - \_\_rhs.base())
- template<typename \_Iterator, typename \_Container >  
constexpr \_\_normal\_iterator< \_Iterator, \_Container >::difference\_type **gnu\_cxx::operator-** (const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_rhs) noexcept
- template<typename \_IteratorL, typename \_IteratorR >  
constexpr auto **std::operator-** (const move\_iterator< \_IteratorL > &\_\_x, const move\_iterator< \_IteratorR > &\_\_y) -> decltype(\_\_x.base() - \_\_y.base())
- template<typename \_Iterator >  
constexpr bool **std::operator<** (const reverse\_iterator< \_Iterator > &\_\_x, const reverse\_iterator< \_Iterator > &\_\_y)
- template<typename \_IteratorL, typename \_IteratorR >  
constexpr bool **std::operator<** (const reverse\_iterator< \_IteratorL > &\_\_x, const reverse\_iterator< \_IteratorR > &\_\_y)
- template<typename \_IteratorL, typename \_IteratorR, typename \_Container >  
bool **gnu\_cxx::operator<** (const \_\_normal\_iterator< \_IteratorL, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_IteratorR, \_Container > &\_\_rhs) noexcept
- template<typename \_Iterator, typename \_Container >  
constexpr bool **gnu\_cxx::operator<** (const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_lhs, const \_\_normal\_iterator< \_Iterator, \_Container > &\_\_rhs) noexcept
- template<typename \_IteratorL, typename \_IteratorR >  
constexpr bool **std::operator<** (const move\_iterator< \_IteratorL > &\_\_x, const move\_iterator< \_IteratorR > &\_\_y)
- template<typename \_Iterator >  
constexpr bool **std::operator<** (const move\_iterator< \_Iterator > &\_\_x, const move\_iterator< \_Iterator > &\_\_y)
- template<typename \_Iterator >  
constexpr bool **std::operator<=** (const reverse\_iterator< \_Iterator > &\_\_x, const reverse\_iterator< \_Iterator > &\_\_y)

- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`  
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator<= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`constexpr bool gnu_cxx::operator<= (const __normal_iterator< _Iterator, _Container > &__lhs, const __↵`  
`normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator<= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &↵`  
`__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator== (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`  
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`  
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`constexpr bool gnu_cxx::operator== (const __normal_iterator< _IteratorL, _Container > &__lhs, const __↵`  
`normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`constexpr bool gnu_cxx::operator== (const __normal_iterator< _Iterator, _Container > &__lhs, const __↵`  
`normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator== (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator== (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`  
`&__y)`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator> (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR`  
`> &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool gnu_cxx::operator> (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_↵`  
`iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`constexpr bool gnu_cxx::operator> (const __normal_iterator< _Iterator, _Container > &__lhs, const __↵`  
`normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR >`  
`&__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator>= (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator >`  
`&__y)`

- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator>= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)`
- `template<typename _IteratorL, typename _IteratorR, typename _Container >`  
`bool __gnu_cxx::operator>= (const __normal_iterator< _IteratorL, _Container > &__lhs, const __normal_iterator< _IteratorR, _Container > &__rhs) noexcept`
- `template<typename _Iterator, typename _Container >`  
`constexpr bool __gnu_cxx::operator>= (const __normal_iterator< _Iterator, _Container > &__lhs, const __normal_iterator< _Iterator, _Container > &__rhs) noexcept`
- `template<typename _IteratorL, typename _IteratorR >`  
`constexpr bool std::operator>= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)`
- `template<typename _Iterator >`  
`constexpr bool std::operator>= (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)`

### 5.576.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file implements `reverse_iterator`, `back_insert_iterator`, `front_insert_iterator`, `insert_iterator`, `__normal_iterator`, and their supporting functions and overloaded operators.

## 5.577 `std_iterator.h` File Reference

### Namespaces

- [`\_\_gnu\_debug`](#)

### Functions

- `template<typename _Iterator, typename _Sequence >`  
`std::reverse\_iterator< _Iterator > __gnu_debug::__base (const std::reverse\_iterator< \_Safe\_iterator< \_Iterator, \_Sequence, std::random\_access\_iterator\_tag > > &__it)`
- `template<typename _Iterator >`  
`auto __gnu_debug::__base (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(\_\_base(\_\_it.base())))`
- `template<typename _Iterator, typename _Size >`  
`bool __gnu_debug::__can_advance (const std::reverse\_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator, typename _Size >`  
`bool __gnu_debug::__can_advance (const std::move\_iterator< _Iterator > &__it, _Size __n)`
- `template<typename _Iterator >`  
`\_Distance\_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator< _Iterator > &__last)`
- `template<typename _Iterator >`  
`\_Distance\_traits< _Iterator >::__type __gnu_debug::__get_distance (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator< _Iterator > &__last)`

- `template<typename _Iterator >`  
`auto __gnu_debug::__unsafe (const std::reverse\_iterator< _Iterator > &__it) -> decltype(std::__make_↵`  
`reverse_iterator(__unsafe(__it.base())))`
- `template<typename _Iterator >`  
`auto __gnu_debug::__unsafe (const std::move\_iterator< _Iterator > &__it) -> decltype(std::make_move_↵`  
`iterator(__unsafe(__it.base())))`
- `template<typename _Iterator >`  
`bool __gnu_debug::__valid_range (const std::reverse\_iterator< _Iterator > &__first, const std::reverse\_iterator<`  
`_Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`
- `template<typename _Iterator >`  
`bool __gnu_debug::__valid_range (const std::move\_iterator< _Iterator > &__first, const std::move\_iterator<`  
`_Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)`

### 5.577.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.578 `std_iterator_base_funcs.h` File Reference

### Classes

- struct [std::\\_List\\_const\\_iterator](#)< \_Tp >
- struct [std::\\_List\\_iterator](#)< \_Tp >

### Namespaces

- [std](#)

### Functions

- `template<typename _InputIterator , typename _Distance >`  
`constexpr void std::\_\_advance (_InputIterator &__i, _Distance __n, input_iterator_tag)`
- `template<typename _BidirectionalIterator , typename _Distance >`  
`constexpr void std::\_\_advance (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)`
- `template<typename _RandomAccessIterator , typename _Distance >`  
`constexpr void std::\_\_advance (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)`
- `template<typename _InputIterator >`  
`constexpr iterator_traits< _InputIterator >::difference_type std::\_\_distance (_InputIterator __first, _InputIterator`  
`__last, input_iterator_tag)`
- `template<typename _RandomAccessIterator >`  
`constexpr iterator_traits< _RandomAccessIterator >::difference_type std::\_\_distance (_RandomAccessIterator`  
`__first, _RandomAccessIterator __last, random_access_iterator_tag)`
- `template<typename _InputIterator , typename _Distance >`  
`constexpr void std::advance (_InputIterator &__i, _Distance __n)`
- `template<typename _InputIterator >`  
`constexpr iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator ↵`  
`__last)`
- `template<typename _InputIterator >`  
`constexpr _InputIterator std::next (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type`  
`__n=1)`
- `template<typename _BidirectionalIterator >`  
`constexpr _BidirectionalIterator std::prev (_BidirectionalIterator __x, typename iterator_traits< _Bidirectional↵`  
`Iterator >::difference_type __n=1)`



## 5.578.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility functions, such as `distance()` and `advance()`.

5.579 `std_iterator_base_types.h` File Reference

## Classes

- struct [std::bidirectional\\_iterator\\_tag](#)
- struct [std::forward\\_iterator\\_tag](#)
- struct [std::input\\_iterator\\_tag](#)
- struct [std::iterator<\\_Category, \\_Tp, \\_Distance, \\_Pointer, \\_Reference>](#)
- struct [std::iterator\\_traits<\\_Iterator>](#)
- struct [std::iterator\\_traits<\\_Iterator>](#)
- struct [std::iterator\\_traits<\\_Tp\\*>](#)
- struct [std::iterator\\_traits<const \\_Tp\\*>](#)
- struct [std::output\\_iterator\\_tag](#)
- struct [std::random\\_access\\_iterator\\_tag](#)

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Iter>`  
using **std::\_\_iterator\_category\_t** = typename `iterator_traits<_Iter>::iterator_category`
- `template<typename _InIter>`  
using **std::RequireInputIter** = `__enable_if_t<is_convertible<__iterator_category_t<_InIter>, input_iterator_tag>::value>`

## Functions

- `template<typename _Iter>`  
`constexpr iterator_traits<_Iter>::iterator_category` [std::\\_\\_iterator\\_category](#) (const \_Iter &)

## 5.579.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

This file contains all of the general iterator-related utility types, such as `iterator_traits` and `struct iterator`.

## 5.580 `std_list.h` File Reference

### Classes

- class [std::\\_List\\_base<\\_Tp, \\_Alloc>](#)
- struct [std::\\_List\\_const\\_iterator<\\_Tp>](#)
- struct [std::\\_List\\_iterator<\\_Tp>](#)
- struct [std::\\_List\\_node<\\_Tp>](#)
- struct [std::\\_\\_detail::\\_List\\_node\\_base](#)
- struct [std::\\_\\_detail::\\_List\\_node\\_header](#)
- class [std::list<\\_Tp, \\_Alloc>](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Functions

- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator!=](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator<](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator<=](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator==](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator>](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[bool std::operator>=](#) (const list<\_Tp, \_Alloc> &\_\_x, const list<\_Tp, \_Alloc> &\_\_y)
- [template<typename \\_Tp, typename \\_Alloc>](#)  
[void std::swap](#) (list<\_Tp, \_Alloc> &\_\_x, list<\_Tp, \_Alloc> &\_\_y) noexcept(*/\*conditional \*/*)

### 5.580.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<list>`.

## 5.581 `std_map.h` File Reference

### Classes

- class [std::map<\\_Key, \\_Tp, \\_Compare, \\_Alloc>](#)
- class [std::multimap<\\_Key, \\_Tp, \\_Compare, \\_Alloc>](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y)`  
`noexcept(/*conditional */)`

## 5.581.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

## 5.582 std\_multimap.h File Reference

## Classes

- class `std::map< \_Key, \_Tp, \_Compare, \_Alloc >`
- class `std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >`

## Namespaces

- [std](#)

## Functions

- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Tp, typename _Compare, typename _Alloc >`  
`void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 5.582.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>`.

### 5.583 stl\_multiset.h File Reference

#### Classes

- class `std::multiset< _Key, _Compare, _Alloc >`
- class `std::set< _Key, _Compare, _Alloc >`

#### Namespaces

- `std`

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

### 5.583.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

## 5.584 `std_numeric.h` File Reference

### Namespaces

- `std`

### Macros

- `#define _GLIBCXX_MOVE_IF_20(_E)`

## Functions

- `template<typename _InputIterator, typename _Tp >`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)`
- `template<typename _InputIterator, typename _Tp, typename _BinaryOperation >`  
`constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↵  
binary_op)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↵  
__result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`constexpr _OutputIterator std::adjacent\_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↵  
__result, _BinaryOperation __binary_op)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp >`  
`constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp  
__init)`
- `template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _BinaryOperation2`  
`>`  
`constexpr _Tp std::inner\_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp  
__init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)`
- `template<typename _ForwardIterator, typename _Tp >`  
`constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)`
- `template<typename _InputIterator, typename _OutputIterator >`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)`
- `template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation >`  
`constexpr _OutputIterator std::partial\_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result,  
_BinaryOperation __binary_op)`

### 5.584.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<numeric>`.

## 5.585 `std_pair.h` File Reference

### Classes

- struct [std::pair<\\_T1, \\_T2>](#)
- struct [std::piecewise\\_construct\\_t](#)

### Namespaces

- [std](#)

### Variables

- `constexpr piecewise_construct_t std::piecewise\_construct`

## 5.585.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

5.586 `std_queue.h` File Reference

## Classes

- class `std::priority_queue<_Tp, _Sequence, _Compare >`
- class `std::queue<_Tp, _Sequence >`

## Namespaces

- `std`

## Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator< (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const queue<_Tp, _Seq > &__x, const queue<_Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`enable_if<__is_swappable<_Seq>::value >::type std::swap (queue<_Tp, _Seq > &__x, queue<_Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Tp, typename _Sequence, typename _Compare >`  
`enable_if<__and<__is_swappable<_Sequence>, __is_swappable<_Compare>>::value >::type std::↔  
::swap (priority_queue<_Tp, _Sequence, _Compare > &__x, priority_queue<_Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))`

## 5.586.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<queue>`.

5.587 `std_raw_storage_iter.h` File Reference

## Classes

- class `std::raw_storage_iterator<_OutputIterator, _Tp >`

## Namespaces

- [std](#)

### 5.587.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.588 `std_relops.h` File Reference

## Namespaces

- [std](#)
- [std::rel\\_ops](#)

## Functions

- `template<class _Tp >`  
`bool std::rel\_ops::operator!= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator<= (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator> (const _Tp &__x, const _Tp &__y)`
- `template<class _Tp >`  
`bool std::rel\_ops::operator>= (const _Tp &__x, const _Tp &__y)`

### 5.588.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<utility>`.

Inclusion of this file has been removed from all of the other STL headers for safety reasons, except `std_↵utility.h`. For more information, see the thread of about twenty messages starting with <http://gcc.gnu.↵org/ml/libstdc++/2001-01/msg00223.html>, or [http://gcc.gnu.org/onlinedocs/libstdc++/faq.↵html#faq.ambiguous\\_overloads](http://gcc.gnu.org/onlinedocs/libstdc++/faq.↵html#faq.ambiguous_overloads)

Short summary: the `rel_ops` operators should be avoided for the present.

## 5.589 `std_set.h` File Reference

## Classes

- class `std::multiset< _Key, _Compare, _Alloc >`
- class `std::set< _Key, _Compare, _Alloc >`



## Namespaces

- [std](#)

## Functions

- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)`
- `template<typename _Key, typename _Compare, typename _Alloc >`  
`void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)`

## 5.589.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<set>`.

5.590 `std_stack.h` File Reference

## Classes

- class [std::stack< \\_Tp, \\_Sequence >](#)

## Namespaces

- [std](#)

## Functions

- `template<typename _Tp, typename _Seq >`  
`bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)`
- `template<typename _Tp, typename _Seq >`  
`enable_if< __is_swappable< _Seq >::value >::type std::swap (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))`

#### 5.590.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<stack>`.

### 5.591 `std_tempbuf.h` File Reference

#### Classes

- class [std::\\_Temporary\\_buffer<\\_ForwardIterator, \\_Tp>](#)

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### Functions

- `template<typename _Tp>`  
`void std::\_\_detail::\_\_return\_temporary\_buffer (_Tp *__p, size_t __len)`
- `template<typename _Pointer, typename _ForwardIterator>`  
`void std::\_\_uninitialized\_construct\_buf (_Pointer __first, _Pointer __last, _ForwardIterator __seed)`
- `template<typename _Tp>`  
`pair<_Tp *, ptrdiff_t> std::get\_temporary\_buffer (ptrdiff_t __len) noexcept`
- `template<typename _Tp>`  
`void std::return\_temporary\_buffer (_Tp *__p)`

#### 5.591.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

### 5.592 `std_tree.h` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define \_\_cpp\_lib\_generic\_associative\_lookup`

## Typedefs

- `template<typename _Cmp, typename _SfinalType >`  
`using std::__has_is_transparent_t = typename __has_is_transparent< _Cmp, _SfinalType >::type`

## Enumerations

- `enum _Rb_tree_color { _S_red, _S_black }`

## Functions

- `unsigned int std::_Rb_tree_black_count (const _Rb_tree_node_base * __node, const _Rb_tree_node_base * __root) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_decrement (_Rb_tree_node_base * __x) throw ()`
- `const _Rb_tree_node_base * std::_Rb_tree_decrement (const _Rb_tree_node_base * __x) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_increment (_Rb_tree_node_base * __x) throw ()`
- `const _Rb_tree_node_base * std::_Rb_tree_increment (const _Rb_tree_node_base * __x) throw ()`
- `void std::_Rb_tree_insert_and_rebalance (const bool __insert_left, _Rb_tree_node_base * __x, _Rb_tree_node_base * __p, _Rb_tree_node_base & __header) throw ()`
- `_Rb_tree_node_base * std::_Rb_tree_rebalance_for_erase (_Rb_tree_node_base * const __z, _Rb_tree_node_base & __header) throw ()`
- `template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc >`  
`void std::swap (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > & __x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > & __y)`

### 5.592.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<map>` or `<set>`.

## 5.593 `std_uninitialized.h` File Reference

### Namespaces

- [`std`](#)

### Functions

- `template<typename _InputIterator, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)`
- `template<typename _InputIterator, typename _Size, typename _ForwardIterator >`  
`_ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)`
- `template<typename _ForwardIterator, typename _Tp >`  
`void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp & __x)`
- `template<typename _ForwardIterator, typename _Size, typename _Tp >`  
`_ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp & __x)`

### 5.593.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.594 `std_vector.h` File Reference

### Classes

- struct [std::\\_Vector\\_base<\\_Tp, \\_Alloc>](#)
- class [std::vector<\\_Tp, \\_Alloc>](#)

### Namespaces

- [std](#)

### Macros

- `#define GLIBCXX_ASAN_ANNOTATE_BEFORE_DEALLOC`
- `#define GLIBCXX_ASAN_ANNOTATE_GREW(n)`
- `#define GLIBCXX_ASAN_ANNOTATE_GROW(n)`
- `#define GLIBCXX_ASAN_ANNOTATE_REINIT`
- `#define GLIBCXX_ASAN_ANNOTATE_SHRINK(n)`

### Functions

- `template<typename _Tp, typename _Alloc>`  
`bool std::operator!= (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator< (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator<= (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator== (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator> (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`bool std::operator>= (const vector<_Tp, _Alloc> &__x, const vector<_Tp, _Alloc> &__y)`
- `template<typename _Tp, typename _Alloc>`  
`void std::swap (vector<_Tp, _Alloc> &__x, vector<_Tp, _Alloc> &__y) noexcept(/*conditional */)`

### 5.594.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 5.595 stop\_token File Reference

### 5.595.1 Detailed Description

This is a Standard C++ Library header.

## 5.596 stream\_iterator.h File Reference

### Classes

- class [std::istream\\_iterator< \\_Tp, \\_CharT, \\_Traits, \\_Dist >](#)
- class [std::ostream\\_iterator< \\_Tp, \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### 5.596.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.597 streambuf File Reference

### Classes

- class [std::basic\\_streambuf< \\_CharT, \\_Traits >](#)

### Namespaces

- [std](#)

### Macros

- `#define _GLIBXX_STREAMBUF`
- `#define _IsUnused`

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof` (`basic_streambuf< _CharT, _Traits > *`, `basic_streambuf< _CharT, _Traits > *`, `bool &`)
- `template<>`  
`streamsize std::__copy_streambufs_eof` (`basic_streambuf< char > *__sbin`, `basic_streambuf< char > *__s`  
`bout`, `bool &__ineof`)
- `template<>`  
`streamsize std::__copy_streambufs_eof` (`basic_streambuf< wchar_t > *__sbin`, `basic_streambuf< wchar_t >`  
`*__sbout`, `bool &__ineof`)

### 5.597.1 Detailed Description

This is a Standard C++ Library header.

## 5.598 streambuf.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define __STREAMBUF_TCC`

## Functions

- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs` (`basic_streambuf< _CharT, _Traits > *__sbin`, `basic_streambuf< _CharT,`  
`_Traits > *__sbout`)
- `template<typename _CharT, typename _Traits >`  
`streamsize std::__copy_streambufs_eof` (`basic_streambuf< _CharT, _Traits > *`, `basic_streambuf< _CharT,`  
`_Traits > *`, `bool &`)

### 5.598.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<streambuf>`.

## 5.599 streambuf\_iterator.h File Reference

### Classes

- class [std::istreambuf\\_iterator< \\_CharT, \\_Traits >](#)
- class [std::ostreambuf\\_iterator< \\_CharT, \\_Traits >](#)

## Namespaces

- [std](#)

## Functions

- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__↵`  
`copy_move_a2 (_CharT * __first, _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__↵`  
`copy_move_a2 (const _CharT * __first, const _CharT * __last, ostreambuf_iterator< _CharT > __result)`
- `template<bool _IsMove, typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type std::__copy_move_a2`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT * __result)`
- `template<typename _CharT, typename _Size >`  
`__enable_if_t< __is_char< _CharT >::__value, _CharT * > std::__copy_n_a (istreambuf_iterator< _CharT >`  
`__it, _Size __n, _CharT * __result)`
- `template<typename _CharT, typename _Distance >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type std::__advance (istreambuf_iterator<`  
`_CharT > & __i, _Distance __n)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type std::__copy`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT >`  
`__result)`
- `template<typename _CharT >`  
`__gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type std::__find`  
`(istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT & __val)`
- `template<typename _CharT, typename _Traits >`  
`bool std::__operator!= (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`  
`_Traits > & __b)`
- `template<typename _CharT, typename _Traits >`  
`bool std::__operator== (const istreambuf_iterator< _CharT, _Traits > & __a, const istreambuf_iterator< _CharT,`  
`_Traits > & __b)`

## 5.599.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<iterator>`.

## 5.600 string File Reference

## Macros

- `#define _GLIBCXX_STRING`

## 5.600.1 Detailed Description

This is a Standard C++ Library header.

## 5.601 string File Reference

### Classes

- class [\\_\\_gnu\\_debug::basic\\_string<\\_CharT, \\_Traits, \\_Allocator>](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)

### Macros

- `#define __glibcxx_check_string_constructor(_Str)`
- `#define __glibcxx_check_string_n_constructor(_Str, _Size)`
- `#define _GLIBCXX_DEBUG_STRING`
- `#define _GLIBCXX_DEBUG_VERIFY_STR_COND_AT(_Cond, _File, _Line, _Func)`

### Typedefs

- `typedef basic_string< char > \_\_gnu\_debug::string`
- `typedef basic_string< wchar_t > \_\_gnu\_debug::wstring`

### Functions

- `template<typename _CharT, typename _Integer>  
const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s, _Integer __n, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT>  
const _CharT * \_\_gnu\_debug::\_\_check\_string (const _CharT * __s, const char * __file, unsigned int __line, const char * __function)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
std::basic\_istream< _CharT, _Traits > & \_\_gnu\_debug::getline (std::basic\_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Allocator > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
std::basic\_istream< _CharT, _Traits > & \_\_gnu\_debug::getline (std::basic\_istream< _CharT, _Traits > & __is, basic_string< _CharT, _Traits, _Allocator > & __str)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
bool \_\_gnu\_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
bool \_\_gnu\_debug::operator!= (const _CharT * __lhs, const basic_string< _CharT, _Traits, _Allocator > & __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
bool \_\_gnu\_debug::operator!= (const basic_string< _CharT, _Traits, _Allocator > & __lhs, const _CharT * __↵  
rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator>  
basic_string< _CharT, _Traits, _Allocator > > \_\_gnu\_debug::operator+ (const basic_string< _CharT, _Traits, ↵  
_Allocator > & __lhs, const basic_string< _CharT, _Traits, _Allocator > & __rhs)`



- [illegible]

- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`bool __gnu_debug::operator>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__`  
`__rhs)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`std::basic\_istream< _CharT, _Traits > & __gnu_debug::operator>> (std::basic\_istream< _CharT, _Traits >`  
`&__is, basic_string< _CharT, _Traits, _Allocator > &__str)`
- `template<typename _CharT, typename _Traits, typename _Allocator >`  
`void __gnu_debug::swap (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits,`  
`_Allocator > &__rhs)`

#### 5.601.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

### 5.602 string File Reference

#### Namespaces

- [std](#)
- [std::experimental](#)

#### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING`

#### Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Up >`  
`void std::experimental::fundamentals_v2::erase (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up`  
`&__value)`
- `template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate >`  
`void std::experimental::fundamentals_v2::erase_if (basic_string< _CharT, _Traits, _Alloc > &__cont, __`  
`Predicate __pred)`

#### 5.602.1 Detailed Description

This is a TS C++ Library header.

### 5.603 string\_conversions.h File Reference

#### Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _TRet , typename _Ret = _TRet, typename _CharT , typename... _Base>  
_Ret gnu_cxx::__stoa (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const  
_CharT *__str, std::size_t *__idx, _Base... __base)`
- `template<typename _String , typename _CharT = typename _String::value_type>  
_String gnu_cxx::__to_xstring (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)`

### 5.603.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.604 string\_view File Reference

### Classes

- class [std::experimental::fundamentals\\_v1::basic\\_string\\_view< \\_CharT, \\_Traits >](#)
- struct [std::hash< \\_Tp >](#)

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_string_view`
- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW`

### Typedefs

- using `std::experimental::fundamentals_v1::string_view` = `basic_string_view< char >`
- using `std::experimental::fundamentals_v1::u16string_view` = `basic_string_view< char16_t >`
- using `std::experimental::fundamentals_v1::u32string_view` = `basic_string_view< char32_t >`
- using `std::experimental::fundamentals_v1::wstring_view` = `basic_string_view< wchar_t >`

## Functions

- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator!= (__type_identity_t< basic_string_view< _↵`  
`CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `constexpr basic_string_view< char > std::experimental::literals::string_view_literals::operator""sv (const`  
`char *__str, size_t __len) noexcept`
- `constexpr basic_string_view< wchar_t > std::experimental::literals::string_view_literals::operator""sv`  
`(const wchar_t *__str, size_t __len) noexcept`
- `constexpr basic_string_view< char16_t > std::experimental::literals::string_view_literals::operator""sv`  
`(const char16_t *__str, size_t __len) noexcept`
- `constexpr basic_string_view< char32_t > std::experimental::literals::string_view_literals::operator""sv`  
`(const char32_t *__str, size_t __len) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator< (__type_identity_t< basic_string_view< _↵`  
`CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`basic_ostream< _CharT, _Traits > & std::experimental::fundamentals_v1::operator<< (basic_ostream< _↵`  
`_CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator<= (__type_identity_t< basic_string_view< _↵`  
`_CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator== (__type_identity_t< basic_string_view< _↵`  
`CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`

- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator> (__type_identity_t< basic_string_view< _↵`  
`CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (basic_string_view< _CharT, _Traits > __x,`  
`basic_string_view< _CharT, _Traits > __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (basic_string_view< _CharT, _Traits > __x,`  
`__type_identity_t< basic_string_view< _CharT, _Traits >> __y) noexcept`
- `template<typename _CharT, typename _Traits >`  
`constexpr bool std::experimental::fundamentals_v1::operator>= (__type_identity_t< basic_string_view< ↵`  
`_CharT, _Traits >> __x, basic_string_view< _CharT, _Traits > __y) noexcept`

#### 5.604.1 Detailed Description

This is a TS C++ Library header.

## 5.605 string\_view.tcc File Reference

### Macros

- `#define _GLIBCXX_STRING_VIEW_TCC`

#### 5.605.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string_view>`.

## 5.606 string\_view.tcc File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_STRING_VIEW_TCC`

#### 5.606.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<experimental/string_view>`.

### 5.607 `stringfwd.h` File Reference

#### Classes

- class `std::basic_string<_CharT, _Traits, _Alloc>`
- struct `std::char_traits<_CharT>`

#### Namespaces

- `std`

#### Typedefs

- typedef `basic_string< char >` `std::string`
- typedef `basic_string< char16_t >` `std::u16string`
- typedef `basic_string< char32_t >` `std::u32string`
- typedef `basic_string< wchar_t >` `std::wstring`

#### 5.607.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<string>`.

### 5.608 `stringstream` File Reference

#### Namespaces

- `std`

#### 5.608.1 Detailed Description

This is a Standard C++ Library header.

### 5.609 `synth_access_traits.hpp` File Reference

#### Classes

- struct `__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >`

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC`
- `#define PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC`

## 5.609.1 Detailed Description

Contains an implementation class for a patricia tree.

5.610 `system_error` File Reference

## Classes

- class [std::\\_V2::error\\_category](#)
- struct [std::error\\_code](#)
- struct [std::error\\_condition](#)
- struct [std::hash< error\\_code >](#)
- struct [std::is\\_error\\_code\\_enum< \\_Tp >](#)
- struct [std::is\\_error\\_condition\\_enum< \\_Tp >](#)
- class [std::system\\_error](#)

## Namespaces

- [std](#)

## Macros

- `#define _GLIBCXX_SYSTEM_ERROR`

## Functions

- `const error_category & std::\_V2::generic\_category () noexcept`
- `const error_category & std::\_V2::system\_category () noexcept`

## Variables

- error\_code `std::make_error_code` (errc) noexcept

### 5.610.1 Detailed Description

This is a Standard C++ Library header.

## 5.611 `system_error` File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_SYSTEM_ERROR`

### Variables

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_error_code_enum_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_error_condition_enum_v`

### 5.611.1 Detailed Description

This is a TS C++ Library header.

## 5.612 `tag_and_trait.hpp` File Reference

### Classes

- `struct \_\_gnu\_pbds::associative\_tag`
- `struct \_\_gnu\_pbds::basic\_branch\_tag`
- `struct \_\_gnu\_pbds::basic\_hash\_tag`
- `struct \_\_gnu\_pbds::basic\_invalidation\_guarantee`
- `struct \_\_gnu\_pbds::binary\_heap\_tag`
- `struct \_\_gnu\_pbds::binomial\_heap\_tag`
- `struct \_\_gnu\_pbds::cc\_hash\_tag`
- `struct \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_Tl >`
- `struct \_\_gnu\_pbds::container\_tag`
- `struct \_\_gnu\_pbds::container\_traits< Cntnr >`
- `struct \_\_gnu\_pbds::container\_traits\_base< \_Tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< binary\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< binomial\_heap\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< cc\_hash\_tag >`
- `struct \_\_gnu\_pbds::container\_traits\_base< gp\_hash\_tag >`



- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#)
- [struct \\_\\_gnu\\_pbds::gp\\_hash\\_tag](#)
- [struct \\_\\_gnu\\_pbds::list\\_update\\_tag](#)
- [struct \\_\\_gnu\\_pbds::null\\_node\\_update< \\_Tp1, \\_Tp2, \\_Tp3, \\_Tp4 >](#)
- [struct \\_\\_gnu\\_pbds::null\\_type](#)
- [struct \\_\\_gnu\\_pbds::ov\\_tree\\_tag](#)
- [struct \\_\\_gnu\\_pbds::pairing\\_heap\\_tag](#)
- [struct \\_\\_gnu\\_pbds::pat\\_trie\\_tag](#)
- [struct \\_\\_gnu\\_pbds::point\\_invalidation\\_guarantee](#)
- [struct \\_\\_gnu\\_pbds::priority\\_queue\\_tag](#)
- [struct \\_\\_gnu\\_pbds::range\\_invalidation\\_guarantee](#)
- [struct \\_\\_gnu\\_pbds::rb\\_tree\\_tag](#)
- [struct \\_\\_gnu\\_pbds::rc\\_binomial\\_heap\\_tag](#)
- [struct \\_\\_gnu\\_pbds::sequence\\_tag](#)
- [struct \\_\\_gnu\\_pbds::splay\\_tree\\_tag](#)
- [struct \\_\\_gnu\\_pbds::string\\_tag](#)
- [struct \\_\\_gnu\\_pbds::thin\\_heap\\_tag](#)
- [struct \\_\\_gnu\\_pbds::tree\\_tag](#)
- [struct \\_\\_gnu\\_pbds::trie\\_tag](#)
- [struct \\_\\_gnu\\_pbds::trivial\\_iterator\\_tag](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Typedefs

- [typedef void \\_\\_gnu\\_pbds::trivial\\_iterator\\_difference\\_type](#)

### 5.612.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

## 5.613 tags.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::balanced\\_tag](#)
- struct [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#)
- struct [\\_\\_gnu\\_parallel::default\\_parallel\\_tag](#)
- struct [\\_\\_gnu\\_parallel::equal\\_split\\_tag](#)
- struct [\\_\\_gnu\\_parallel::exact\\_tag](#)
- struct [\\_\\_gnu\\_parallel::find\\_tag](#)
- struct [\\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#)
- struct [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#)
- struct [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#)
- struct [\\_\\_gnu\\_parallel::parallel\\_tag](#)
- struct [\\_\\_gnu\\_parallel::quicksort\\_tag](#)
- struct [\\_\\_gnu\\_parallel::sampling\\_tag](#)
- struct [\\_\\_gnu\\_parallel::sequential\\_tag](#)
- struct [\\_\\_gnu\\_parallel::unbalanced\\_tag](#)

### Namespaces

- [\\_\\_gnu\\_parallel](#)

#### 5.613.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

## 5.614 tgmath.h File Reference

### Macros

- `#define \_GLIBCXX\_TGMATH\_H`

#### 5.614.1 Detailed Description

This is a Standard C++ Library header.

## 5.615 thin\_heap.hpp File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_ASSERT_AUX_NULL(X)`
- `#define PB_DS_ASSERT_NODE_CONSISTENT(_Node, _Bool)`
- `#define PB_DS_BASE_T_P`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## Enumerations

- `enum { num_distinct_rank_bounds }`

## Variables

- `static const std::size_t __gnu_pbds::detail::g_a_rank_bounds [num_distinct_rank_bounds]`

## 5.615.1 Detailed Description

Contains an implementation class for a thin heap.

## 5.616 thread File Reference

## Classes

- struct [std::hash< thread::id >](#)
- class [std::thread::id](#)
- class [std::thread](#)

## Namespaces

- [std](#)
- [std::this\\_thread](#)

## Macros

- `#define _GLIBCXX_THREAD`

## Functions

- void **std::this\_thread::sleep\_for** (chrono::seconds, chrono::nanoseconds)
- thread::id **std::this\_thread::get\_id** () noexcept
- bool **std::operator!=** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator<** (thread::id \_\_x, thread::id \_\_y) noexcept
- template<class \_CharT, class \_Traits >  
basic\_ostream< \_CharT, \_Traits > & **std::operator<<** (basic\_ostream< \_CharT, \_Traits > &\_\_out, thread::id \_\_id)
- bool **std::operator<=** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator==** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator>** (thread::id \_\_x, thread::id \_\_y) noexcept
- bool **std::operator>=** (thread::id \_\_x, thread::id \_\_y) noexcept
- template<typename \_Rep, typename \_Period >  
void **std::this\_thread::sleep\_for** (const chrono::duration< \_Rep, \_Period > &\_\_rtime)
- template<typename \_Clock, typename \_Duration >  
void **std::this\_thread::sleep\_until** (const chrono::time\_point< \_Clock, \_Duration > &\_\_atime)
- void **std::swap** (thread &\_\_x, thread &\_\_y) noexcept
- void **std::this\_thread::yield** () noexcept

### 5.616.1 Detailed Description

This is a Standard C++ Library header.

## 5.617 throw\_allocator.h File Reference

### Classes

- struct [\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::annotate\\_base](#)
- struct [\\_\\_gnu\\_cxx::condition\\_base](#)
- struct [\\_\\_gnu\\_cxx::forced\\_error](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_limit >](#)
- struct [std::hash< \\_\\_gnu\\_cxx::throw\\_value\\_random >](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#)
- struct [\\_\\_gnu\\_cxx::random\\_condition](#)
- class [\\_\\_gnu\\_cxx::throw\\_allocator\\_base< \\_Tp, \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_limit< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_allocator\\_random< \\_Tp >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_base< \\_Cond >](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_limit](#)
- struct [\\_\\_gnu\\_cxx::throw\\_value\\_random](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

## Functions

- `void __gnu_cxx::__throw_forced_error ()`
- `template<typename _Tp, typename _Cond >  
bool __gnu_cxx::operator!= (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond >  
throw_value_base< _Cond > __gnu_cxx::operator* (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond >  
throw_value_base< _Cond > __gnu_cxx::operator+ (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond >  
throw_value_base< _Cond > __gnu_cxx::operator- (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Cond >  
bool __gnu_cxx::operator< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `std::ostream & __gnu_cxx::operator<< (std::ostream &os, const annotate_base &__b)`
- `template<typename _Cond >  
bool __gnu_cxx::operator== (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)`
- `template<typename _Tp, typename _Cond >  
bool __gnu_cxx::operator== (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)`
- `template<typename _Cond >  
void __gnu_cxx::swap (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)`

## 5.617.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains two exception-generating types (throw\_value, throw\_allocator) intended to be used as value and allocator types while testing exception safety in templated containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced\_exception\_error.

## 5.618 time\_members.h File Reference

## Namespaces

- [std](#)

#### 5.618.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<locale>`.

### 5.619 `trace_fn_imps.hpp` File Reference

#### 5.619.1 Detailed Description

Contains an implementation class for a `binary_heap`.

### 5.620 `trace_fn_imps.hpp` File Reference

#### 5.620.1 Detailed Description

Contains implementations of `cc_ht_map_`'s trace-mode functions.

### 5.621 `trace_fn_imps.hpp` File Reference

#### 5.621.1 Detailed Description

Contains implementations of `gp_ht_map_`'s trace-mode functions.

### 5.622 `trace_fn_imps.hpp` File Reference

#### 5.622.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

### 5.623 `trace_fn_imps.hpp` File Reference

#### 5.623.1 Detailed Description

Contains implementations of `lu_map_`.

### 5.624 `trace_fn_imps.hpp` File Reference

#### 5.624.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 5.625 `trace_fn_imps.hpp` File Reference

### 5.625.1 Detailed Description

Contains an implementation for `rc_binomial_heap_`.

## 5.626 `trace_fn_imps.hpp` File Reference

### 5.626.1 Detailed Description

Contains an implementation class for `left_child_next_sibling_heap_`.

## 5.627 `traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.627.1 Detailed Description

Contains an implementation for `bin_search_tree_`.

## 5.628 `traits.hpp` File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Data, Cmp\\_Fn, Node\\_Update, Tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Data, \\_ATraits, Node\\_Update, Tag, \\_Alloc >](#)

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_DEBUG_VERIFY(_Cond)`

#### 5.628.1 Detailed Description

Contains an implementation class for tree-like classes.

### 5.629 traits.hpp File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.629.1 Detailed Description

Contains an implementation class for ov\_tree\_.

### 5.630 traits.hpp File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, Mapped, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::trie\\_traits< Key, null\\_type, \\_ATraits, Node\\_Update, pat\\_trie\\_tag, \\_Alloc >](#)

#### Namespaces

- [\\_\\_gnu\\_pbds](#)

#### 5.630.1 Detailed Description

Contains an implementation class for pat\_trie\_.

### 5.631 traits.hpp File Reference

#### Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >](#)



## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.631.1 Detailed Description

Contains an implementation for rb\_tree\_.

## 5.632 traits.hpp File Reference

## Classes

- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)
- struct [\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, splay\\_tree\\_tag, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## 5.632.1 Detailed Description

Contains an implementation for splay\_tree\_.

## 5.633 tree\_policy.hpp File Reference

## Classes

- class [\\_\\_gnu\\_pbds::tree\\_order\\_statistics\\_node\\_update< Node\\_Cltr, Node\\_Itr, Cmp\\_Fn, \\_Alloc >](#)

## Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_BRANCH_POLICY_BASE`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

## 5.633.1 Detailed Description

Contains tree-related policies.

## 5.634 `tree_trace_base.hpp` File Reference

### 5.634.1 Detailed Description

Contains tree-related policies.

## 5.635 `trie_policy.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::trie\\_order\\_statistics\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- class [\\_\\_gnu\\_pbds::trie\\_prefix\\_search\\_node\\_update](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >
- struct [\\_\\_gnu\\_pbds::trie\\_string\\_access\\_traits](#)< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_CLASS_T_DEC`
- `#define PB_DS_TRIE_POLICY_BASE`

### 5.635.1 Detailed Description

Contains trie-related policies.

## 5.636 `trie_policy_base.hpp` File Reference

### Classes

- class [\\_\\_gnu\\_pbds::detail::trie\\_policy\\_base](#)< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >

### Namespaces

- [\\_\\_gnu\\_pbds](#)

## Macros

- `#define PB_DS_CLASS_C_DEC`
- `#define PB_DS_CLASS_T_DEC`

### 5.636.1 Detailed Description

Contains an implementation of `trie_policy_base`.

## 5.637 trie\_string\_access\_traits\_imp.hpp File Reference

### 5.637.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

## 5.638 tuple File Reference

## Classes

- struct `std::_Tuple_impl<_Idx, _Elements>`
- struct `std::_Tuple_impl<_Idx, _Head, _Tail...>`
- class `std::tuple<_Elements>`
- class `std::tuple<_Elements>`
- class `std::tuple<_T1, _T2>`
- struct `std::tuple_element<0, tuple<_Head, _Tail...>>`
- struct `std::tuple_element<__i, tuple<_Head, _Tail...>>`
- struct `std::tuple_element<__i, tuple<>>`
- struct `std::tuple_size<tuple<_Elements...>>`
- struct `std::uses_allocator<tuple<_Types...>, _Alloc>`

## Namespaces

- `std`

## Macros

- `#define __cpp_lib_tuples_by_type`
- `#define _GLIBCXX_TUPLE`

## Typedefs

- `template<typename _Tp>`  
using `std::__empty_not_final` = `typename conditional<__is_final(_Tp), false_type, __is_empty_non_tuple<_Tp>>::type`

## Functions

- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr _Head & std::__get_helper ( _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<std::size_t __i, typename _Head, typename... _Tail>`  
`constexpr const _Head & std::__get_helper (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr _Head & std::__get_helper2 ( _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename _Head, size_t __i, typename... _Tail>`  
`constexpr const _Head & std::__get_helper2 (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< _Elements &&... > std::forward_as_tuple ( _Elements &&... __args) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept`
- `template<std::size_t __i, typename... _Elements>`  
`constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp & std::get (tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr _Tp && std::get (tuple< _Types... > &&__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp & std::get (const tuple< _Types... > &__t) noexcept`
- `template<typename _Tp, typename... _Types>`  
`constexpr const _Tp && std::get (const tuple< _Types... > &&__t) noexcept`
- `template<typename... _Elements>`  
`constexpr tuple< typename __decay_and_strip< _Elements >::type... > std::make_tuple ( _Elements &&... __args)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator!= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator< (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator<= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator== (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator> (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _TElements, typename... _UElements>`  
`constexpr bool std::operator>= (const tuple< _TElements... > &__t, const tuple< _UElements... > &__u)`
- `template<typename... _Elements>`  
`constexpr enable_if< __and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename... _Elements>`  
`constexpr enable_if<!__and< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete`

- `template<typename... _Elements>`  
`constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept`
- `template<typename... _Tpls, typename = typename enable_if<__and<__is_tuple_like<_Tpls>...>::value>::type>`  
`constexpr auto std::tuple\_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls... >::__type`

#### Variables

- `constexpr _Swallow_assign std::ignore`

#### 5.638.1 Detailed Description

This is a Standard C++ Library header.

### 5.639 tuple File Reference

#### Namespaces

- [std](#)
- [std::experimental](#)

#### Macros

- `#define __cpp_lib_experimental_tuple`
- `#define _GLIBCXX_EXPERIMENTAL_TUPLE`

#### Functions

- `template<typename _Fn, typename _Tuple, std::size_t... _Idx>`  
`decltype(auto) constexpr std::experimental::fundamentals_v1::__apply_impl (_Fn &&__f, _Tuple &&__t, std::index\_sequence<_Idx... >)`
- `template<typename _Fn, typename _Tuple >`  
`decltype(auto) constexpr std::experimental::fundamentals_v1::apply (_Fn &&__f, _Tuple &&__t)`

#### Variables

- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::tuple_size_v`

#### 5.639.1 Detailed Description

This is a TS C++ Library header.

## 5.640 type\_traits File Reference

### Classes

- struct `std::__add_pointer_helper< _Tp, bool >`
- struct `std::__detector< _Default, _AlwaysVoid, _Op, _Args >`
- struct `std::__detector< _Default, __void_t< _Op< _Args... > >, _Op, _Args... >`
- struct `std::__is_nullptr_t< _Tp >`
- struct `std::add_const< _Tp >`
- struct `std::add_cv< _Tp >`
- struct `std::add_lvalue_reference< _Tp >`
- struct `std::add_rvalue_reference< _Tp >`
- struct `std::add_volatile< _Tp >`
- struct `std::aligned_storage< _Len, _Align >`
- struct `std::aligned_union< _Len, _Types >`
- struct `std::alignment_of< _Tp >`
- struct `std::common_type< _Tp >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- struct `std::conditional< _Cond, _Iftrue, _Iffalse >`
- class `std::decay< _Tp >`
- struct `std::enable_if< bool, _Tp >`
- struct `std::extent< typename, _Uint >`
- struct `std::extent< typename, _Uint >`
- struct `std::has_virtual_destructor< _Tp >`
- struct `std::integral_constant< _Tp, __v >`
- struct `std::is_abstract< _Tp >`
- struct `std::is_arithmetic< _Tp >`
- struct `std::is_array< typename >`
- struct `std::is_assignable< _Tp, _Up >`
- struct `std::is_base_of< _Base, _Derived >`
- struct `std::is_class< _Tp >`
- struct `std::is_compound< _Tp >`
- struct `std::is_const< typename >`
- struct `std::is_const< typename >`
- struct `std::is_constructible< _Tp, _Args >`
- struct `std::is_convertible< _From, _To >`
- struct `std::is_copy_assignable< _Tp >`
- struct `std::is_copy_constructible< _Tp >`
- struct `std::is_default_constructible< _Tp >`
- struct `std::is_destructible< _Tp >`
- struct `std::is_empty< _Tp >`
- struct `std::is_enum< _Tp >`
- struct `std::is_final< _Tp >`
- struct `std::is_floating_point< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_function< _Tp >`
- struct `std::is_fundamental< _Tp >`
- struct `std::is_integral< _Tp >`
- struct `std::is_literal_type< _Tp >`
- struct `std::is_lvalue_reference< typename >`

- struct `std::is_member_function_pointer< _Tp >`
- struct `std::is_member_object_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_member_pointer< _Tp >`
- struct `std::is_move_assignable< _Tp >`
- struct `std::is_move_constructible< _Tp >`
- struct `std::is_nothrow_assignable< _Tp, _Up >`
- struct `std::is_nothrow_constructible< _Tp, _Args >`
- struct `std::is_nothrow_copy_assignable< _Tp >`
- struct `std::is_nothrow_copy_constructible< _Tp >`
- struct `std::is_nothrow_default_constructible< _Tp >`
- struct `std::is_nothrow_destructible< _Tp >`
- struct `std::is_nothrow_move_assignable< _Tp >`
- struct `std::is_nothrow_move_constructible< _Tp >`
- struct `std::is_nothrow_swappable< _Tp >`
- struct `std::is_nothrow_swappable_with< _Tp, _Up >`
- struct `std::is_null_pointer< _Tp >`
- struct `std::is_object< _Tp >`
- struct `std::is_pod< _Tp >`
- struct `std::is_pointer< _Tp >`
- struct `std::is_polymorphic< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_reference< _Tp >`
- struct `std::is_rvalue_reference< typename >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_same< _Tp, _Up >`
- struct `std::is_scalar< _Tp >`
- struct `std::is_standard_layout< _Tp >`
- struct `std::is_swappable< _Tp >`
- struct `std::is_swappable_with< _Tp, _Up >`
- struct `std::is_trivial< _Tp >`
- struct `std::is_trivially_assignable< _Tp, _Up >`
- struct `std::is_trivially_constructible< _Tp, _Args >`
- struct `std::is_trivially_copy_assignable< _Tp >`
- struct `std::is_trivially_copy_constructible< _Tp >`
- struct `std::is_trivially_default_constructible< _Tp >`
- struct `std::is_trivially_destructible< _Tp >`
- struct `std::is_trivially_move_assignable< _Tp >`
- struct `std::is_trivially_move_constructible< _Tp >`
- struct `std::is_union< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_void< _Tp >`
- struct `std::is_volatile< typename >`
- struct `std::make_signed< _Tp >`
- struct `std::make_unsigned< _Tp >`
- struct `std::rank< typename >`
- class `std::reference_wrapper< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_all_extents< _Tp >`
- struct `std::remove_const< _Tp >`
- struct `std::remove_cv< _Tp >`

- struct `std::remove_cv< _Tp >`
- struct `std::remove_extent< _Tp >`
- struct `std::remove_pointer< _Tp >`
- struct `std::remove_reference< _Tp >`
- struct `std::remove_volatile< _Tp >`
- class `std::result_of< _Signature >`
- class `std::tuple< _Elements >`
- struct `std::underlying_type< _Tp >`

## Namespaces

- `std`

## Macros

- `#define __cpp_lib_integral_constant_callable`
- `#define __cpp_lib_is_final`
- `#define __cpp_lib_is_null_pointer`
- `#define __cpp_lib_is_swappable`
- `#define __cpp_lib_result_of_sfinae`
- `#define __cpp_lib_transformation_trait_aliases`
- `#define __cpp_lib_void_t`
- `#define _GLIBCXX_HAS_NESTED_TYPE(_NTYPE)`
- `#define _GLIBCXX_TYPE_TRAITS`

## Typedefs

- `template<bool __v>`  
using `std::__bool_constant` = `integral_constant< bool, __v >`
- `template<typename _Fn, typename... _Args>`  
using `std::__call_is_nothrow` = `__call_is_nothrow< __invoke_result< _Fn, _Args... >, _Fn, _Args... >`
- `template<typename _Tp >`  
using `std::__decay_and_strip` = `__strip_reference_wrapper< __decay_t< _Tp > >`
- `template<typename _Tp >`  
using `std::__decay_t` = `typename decay< _Tp >::type`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
using `std::__detected_or` = `__detector< _Default, void, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
using `std::__detected_or_t` = `typename __detected_or< _Default, _Op, _Args... >::type`
- `template<bool _Cond, typename _Tp = void>`  
using `std::__enable_if_t` = `typename enable_if< _Cond, _Tp >::type`
- `template<typename _ToElementType, typename _FromElementType >`  
using `std::__is_array_convertible` = `is_convertible< _FromElementType(*)[], _ToElementType(*)[] >`
- `template<typename _Tp, typename... _Args>`  
using `std::__is_nothrow_constructible_impl` = `__is_nt_constructible_impl< __is_constructible(_Tp, _Args...), _Tp, _Args... >`
- `template<typename _Tp, typename... _Types>`  
using `std::__is_one_of` = `__or< is_same< _Tp, _Types >... >`



- `template<typename _Tp >`  
`using std::\_\_is\_signed\_integer = __is_one_of< __remove_cv_t< _Tp >, signed char, signed short, signed int, signed long, signed long long >`
- `template<typename _Tp >`  
`using std::\_\_is\_standard\_integer = __or< __is_signed_integer< _Tp >, __is_unsigned_integer< _Tp > >`
- `template<typename _Tp >`  
`using std::\_\_is\_unsigned\_integer = __is_one_of< __remove_cv_t< _Tp >, unsigned char, unsigned short, unsigned int, unsigned long, unsigned long long >`
- `template<typename _Tp >`  
`using std::\_\_remove\_cv\_t = typename remove_cv< _Tp >::type`
- `template<typename _Tp >`  
`using std::\_\_remove\_cvref\_t = typename remove_cv< typename remove_reference< _Tp >::type >::type`
- `template<typename _Tp >`  
`using std::\_\_type\_identity\_t = typename __type_identity< _Tp >::type`
- `template<typename... >`  
`using std::\_\_void\_t = void`
- `template<typename... _Cond>`  
`using std::\_\_Require = __enable_if_t< __and< _Cond... >::value >`
- `template<typename _Tp >`  
`using std::add\_const\_t = typename add_const< _Tp >::type`
- `template<typename _Tp >`  
`using std::add\_cv\_t = typename add_cv< _Tp >::type`
- `template<typename _Tp >`  
`using std::add\_lvalue\_reference\_t = typename add_lvalue_reference< _Tp >::type`
- `template<typename _Tp >`  
`using std::add\_pointer\_t = typename add_pointer< _Tp >::type`
- `template<typename _Tp >`  
`using std::add\_rvalue\_reference\_t = typename add_rvalue_reference< _Tp >::type`
- `template<typename _Tp >`  
`using std::add\_volatile\_t = typename add_volatile< _Tp >::type`
- `template<size_t _Len, size_t _Align = __alignof__(typename __aligned_storage_msa< _Len >::type)>`  
`using std::aligned\_storage\_t = typename aligned_storage< _Len, _Align >::type`
- `template<size_t _Len, typename... _Types>`  
`using std::aligned\_union\_t = typename aligned_union< _Len, _Types... >::type`
- `template<typename... _Tp>`  
`using std::common\_type\_t = typename common_type< _Tp... >::type`
- `template<bool _Cond, typename _Iftrue, typename _Iffalse >`  
`using std::conditional\_t = typename conditional< _Cond, _Iftrue, _Iffalse >::type`
- `template<typename _Tp >`  
`using std::decay\_t = typename decay< _Tp >::type`
- `template<bool _Cond, typename _Tp = void>`  
`using std::enable\_if\_t = typename enable_if< _Cond, _Tp >::type`
- `typedef integral_constant< bool, false > std::false\_type`
- `template<typename _Tp >`  
`using std::make\_signed\_t = typename make_signed< _Tp >::type`
- `template<typename _Tp >`  
`using std::make\_unsigned\_t = typename make_unsigned< _Tp >::type`
- `template<typename _Tp >`  
`using std::remove\_all\_extents\_t = typename remove_all_extents< _Tp >::type`
- `template<typename _Tp >`  
`using std::remove\_const\_t = typename remove_const< _Tp >::type`
- `template<typename _Tp >`  
`using std::remove\_cv\_t = typename remove_cv< _Tp >::type`

- `template<typename _Tp >`  
using `std::remove_extent_t` = `typename remove_extent< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_pointer_t` = `typename remove_pointer< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_reference_t` = `typename remove_reference< _Tp >::type`
- `template<typename _Tp >`  
using `std::remove_volatile_t` = `typename remove_volatile< _Tp >::type`
- `template<typename _Tp >`  
using `std::result_of_t` = `typename result_of< _Tp >::type`
- `typedef integral_constant< bool, true > std::true_type`
- `template<typename _Tp >`  
using `std::underlying_type_t` = `typename underlying_type< _Tp >::type`
- `template<typename... >`  
using `std::void_t` = `void`

## Functions

- `template<typename _Fn, typename _Tp, typename... _Args>`  
constexpr bool `std::__call_is_nt` (`__invoke_memfun_ref`)
- `template<typename _Fn, typename _Tp, typename... _Args>`  
constexpr bool `std::__call_is_nt` (`__invoke_memfun_deref`)
- `template<typename _Fn, typename _Tp >`  
constexpr bool `std::__call_is_nt` (`__invoke_memobj_ref`)
- `template<typename _Fn, typename _Tp >`  
constexpr bool `std::__call_is_nt` (`__invoke_memobj_deref`)
- `template<typename _Fn, typename... _Args>`  
constexpr bool `std::__call_is_nt` (`__invoke_other`)
- `template<typename _Tp, size_t = sizeof(_Tp)>`  
constexpr true\_type `std::__is_complete_or_unbounded` (`__type_identity< _Tp >`)
- `template<typename _TypeIdentity, typename _NestedType = typename _TypeIdentity::type>`  
constexpr `__or_< is_reference< _NestedType >, is_function< _NestedType >, is_void< _NestedType >, __is_array_unknown_bounds< _NestedType > >::type` `std::__is_complete_or_unbounded` (`__TypeIdentity`)
- `template<typename _Tp >`  
`std::is_nullptr_t` is `__null_pointer` `std::__GLIBCXX_DEPRECATED_SUGGEST` ("std::is\_null\_pointer")
- `template<typename _Tp >`  
auto `std::declval` () noexcept -> `decltype(__declval< _Tp >())`
- `template<typename _Tp >`  
constexpr `__Require< __not_< __is_tuple_like< _Tp >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap` (`_Tp &, _Tp &`) noexcept(`__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value`)
- `template<typename _Tp, size_t _Nm>`  
constexpr `__enable_if_t< __is_swappable< _Tp >::value > std::swap` (`_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]`) noexcept(`__is_nothrow_swappable< _Tp >::value`)

## Variables

- `std::is_reference` `std::__GLIBCXX_DEPRECATED_SUGGEST`
- `template<typename _Tp >`  
constexpr bool `std::is_nothrow_swappable_v`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::is\_nothrow\_swappable\_with\_v`
- `template<typename _Tp >`  
`constexpr bool std::is\_swappable\_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::is\_swappable\_with\_v`

#### 5.640.1 Detailed Description

This is a Standard C++ Library header.

#### 5.640.2 Macro Definition Documentation

##### 5.640.2.1 `_GLIBCXX_HAS_NESTED_TYPE`

```
#define _GLIBCXX_HAS_NESTED_TYPE(
 _NTYPE)
```

Use SFINAE to determine if the type `_Tp` has a publicly-accessible member type `_NTYPE`.

Definition at line 2614 of file `type_traits`.

## 5.641 `type_traits` File Reference

### Classes

- struct `std::tr2::\_\_reflection\_typelist< \_Elements >`
- struct `std::tr2::\_\_reflection\_typelist< \_First, \_Rest... >`
- struct `std::tr2::\_\_reflection\_typelist<>`
- struct `std::tr2::bases< \_Tp >`
- struct `std::tr2::direct\_bases< \_Tp >`

### Namespaces

- `std`
- `std::tr2`

### Macros

- `#define \_GLIBCXX\_TR2\_TYPE\_TRAITS`

### 5.641.1 Detailed Description

This is a TR2 C++ Library header.

## 5.642 `type_traits` File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_detect`
- `#define __cpp_lib_experimental_logical_traits`
- `#define __cpp_lib_experimental_type_trait_variable_templates`
- `#define _GLIBCXX_EXPERIMENTAL_TYPE_TRAITS`

### Typedefs

- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_or = std::\_\_detected\_or< _Default, _Op, _Args... >`
- `template<typename _Default, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_or_t = typename detected\_or< _Default, _Op, _Args... >::type`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::detected_t = typename std::\_\_detector< nonesuch, void, _Op, ↵  
_Args... >::type`
- `template<template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected = typename std::\_\_detector< nonesuch, void, _Op,  
_Args... >::value_t`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected_convertible = is_convertible< detected_t< _Op, ↵  
_Args... >, _To >`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`using std::experimental::fundamentals_v2::is_detected_exact = is_same< _Expected, detected_t< _Op, ↵  
_Args... > >`
- `template<typename... >`  
`using std::experimental::fundamentals_v2::void_t = void`

## Variables

- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::alignment_of_v`
- `template<typename... _Bn >`  
`constexpr bool std::experimental::fundamentals_v2::conjunction_v`
- `template<typename... _Bn >`  
`constexpr bool std::experimental::fundamentals_v2::disjunction_v`
- `template<typename _Tp, unsigned _Idx = 0>`  
`constexpr size_t std::experimental::fundamentals_v1::extent_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::has_virtual_destructor_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_abstract_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_arithmetic_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_array_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_assignable_v`
- `template<typename _Base, typename _Derived >`  
`constexpr bool std::experimental::fundamentals_v1::is_base_of_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_class_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_compound_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_const_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_constructible_v`
- `template<typename _From, typename _To >`  
`constexpr bool std::experimental::fundamentals_v1::is_convertible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_destructible_v`
- `template<typename _To, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_convertible_v`
- `template<typename _Expected, template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_exact_v`
- `template<template< typename... > class _Op, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v2::is_detected_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_empty_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_enum_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_final_v`

- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_floating_point_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_function_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_fundamental_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_integral_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_literal_type_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_lvalue_reference_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_function_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_object_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_member_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_move_constructible_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_destructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_nothrow_move_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_null_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_object_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_pod_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_pointer_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_polymorphic_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_reference_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_rvalue_reference_v`

- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_same_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_scalar_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_signed_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_standard_layout_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivial_v`
- `template<typename _Tp, typename _Up >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_assignable_v`
- `template<typename _Tp, typename... _Args>`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copy_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copy_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_copyable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_default_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_destructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_move_assignable_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_trivially_move_constructible_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_union_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_unsigned_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_void_v`
- `template<typename _Tp >`  
`constexpr bool std::experimental::fundamentals_v1::is_volatile_v`
- `template<typename _Pp >`  
`constexpr bool std::experimental::fundamentals_v2::negation_v`
- `template<typename _Tp >`  
`constexpr size_t std::experimental::fundamentals_v1::rank_v`

#### 5.642.1 Detailed Description

This is a TS C++ Library header.

## 5.643 type\_traits.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

## Functions

- `template<typename _Type >`  
`bool __gnu_cxx::__is_null_pointer (_Type *__ptr)`
- `template<typename _Type >`  
`bool __gnu_cxx::__is_null_pointer (_Type)`
- `bool __gnu_cxx::__is_null_pointer (std::nullptr_t)`

### 5.643.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.644 `type_utils.hpp` File Reference

### Namespaces

- [\\_\\_gnu\\_pbds](#)

### Macros

- `#define PB_DS_STATIC_ASSERT(UNIQUE, E)`

### Typedefs

- `typedef std::tr1::integral_constant< int, 0 > __gnu_pbds::detail::false_type`
- `typedef std::tr1::integral_constant< int, 1 > __gnu_pbds::detail::true_type`

### 5.644.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alexandrescu.

## 5.645 `typeindex` File Reference

### Classes

- struct [std::hash< \\_Tp >](#)
- struct [std::hash< type\\_index >](#)
- struct [std::type\\_index](#)

### Namespaces

- [std](#)



## Macros

- `#define _GLIBCXX_TYPEINDEX`

### 5.645.1 Detailed Description

This is a Standard C++ Library header.

## 5.646 `typeinfo` File Reference

## Classes

- class [std::bad\\_cast](#)
- class [std::bad\\_typeid](#)
- class [std::type\\_info](#)

## Namespaces

- [std](#)

## Macros

- `#define __GXX_MERGED_TYPEINFO_NAMES`
- `#define __GXX_TYPEINFO_EQUALITY_INLINE`
- `#define _TYPEINFO`

### 5.646.1 Detailed Description

This is a Standard C++ Library header.

## 5.647 `typelist.h` File Reference

## Namespaces

- [\\_\\_gnu\\_cxx](#)
- [\\_\\_gnu\\_cxx::typelist](#)

## Macros

- `#define _GLIBCXX_TYPELIST_CHAIN1(X0)`
- `#define _GLIBCXX_TYPELIST_CHAIN10(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)`
- `#define _GLIBCXX_TYPELIST_CHAIN11(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)`
- `#define _GLIBCXX_TYPELIST_CHAIN12(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)`
- `#define _GLIBCXX_TYPELIST_CHAIN13(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)`
- `#define _GLIBCXX_TYPELIST_CHAIN14(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)`
- `#define _GLIBCXX_TYPELIST_CHAIN15(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)`
- `#define _GLIBCXX_TYPELIST_CHAIN16(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)`
- `#define _GLIBCXX_TYPELIST_CHAIN17(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)`
- `#define _GLIBCXX_TYPELIST_CHAIN18(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)`
- `#define _GLIBCXX_TYPELIST_CHAIN19(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)`
- `#define _GLIBCXX_TYPELIST_CHAIN2(X0, X1)`
- `#define _GLIBCXX_TYPELIST_CHAIN20(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)`
- `#define _GLIBCXX_TYPELIST_CHAIN3(X0, X1, X2)`
- `#define _GLIBCXX_TYPELIST_CHAIN4(X0, X1, X2, X3)`
- `#define _GLIBCXX_TYPELIST_CHAIN5(X0, X1, X2, X3, X4)`
- `#define _GLIBCXX_TYPELIST_CHAIN6(X0, X1, X2, X3, X4, X5)`
- `#define _GLIBCXX_TYPELIST_CHAIN7(X0, X1, X2, X3, X4, X5, X6)`
- `#define _GLIBCXX_TYPELIST_CHAIN8(X0, X1, X2, X3, X4, X5, X6, X7)`
- `#define _GLIBCXX_TYPELIST_CHAIN9(X0, X1, X2, X3, X4, X5, X6, X7, X8)`

## Functions

- `template<typename Fn, typename Typelist>  
void __gnu_cxx::typelist::apply (Fn &, Typelist)`
- `template<typename Gn, typename Typelist>  
void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)`
- `template<typename Gn, typename TypelistT, typename TypelistV>  
void __gnu_cxx::typelist::apply_generator (Gn &, TypelistT, TypelistV)`
- `template<typename Fn, typename Typelist>  
void __gnu_cxx::typelist::apply_generator (Fn &fn, Typelist)`
- `template<typename Fn, typename TypelistT, typename TypelistV>  
void __gnu_cxx::typelist::apply_generator (Fn &fn, TypelistT, TypelistV)`

### 5.647.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Contains `typelist_chain` definitions. Typelists are an idea by Andrei Alexandrescu.

## 5.648 types.h File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Typedefs

- typedef int64\_t [\\_\\_gnu\\_parallel::\\_CASable](#)
- typedef uint64\_t [\\_\\_gnu\\_parallel::\\_SequenceIndex](#)
- typedef uint16\_t [\\_\\_gnu\\_parallel::\\_ThreadIndex](#)

### Enumerations

- enum [\\_\\_gnu\\_parallel::\\_AlgorithmStrategy](#) { **heuristic**, **force\_sequential**, **force\_parallel** }
- enum [\\_\\_gnu\\_parallel::\\_FindAlgorithm](#) { **GROWING\_BLOCKS**, **CONSTANT\_SIZE\_BLOCKS**, **EQUAL\_SPLIT** }
- enum [\\_\\_gnu\\_parallel::\\_MultiwayMergeAlgorithm](#) { **LOSER\_TREE** }
- enum [\\_\\_gnu\\_parallel::\\_Parallelism](#) { [\\_\\_gnu\\_parallel::sequential](#), [\\_\\_gnu\\_parallel::parallel\\_unbalanced](#), [\\_\\_gnu\\_parallel::parallel\\_balanced](#), [\\_\\_gnu\\_parallel::parallel\\_omp](#), [\\_\\_gnu\\_parallel::parallel\\_omp\\_loop\\_static](#), [\\_\\_gnu\\_parallel::parallel\\_taskqueue](#) }
- enum [\\_\\_gnu\\_parallel::\\_PartialSumAlgorithm](#) { **RECURSIVE**, **LINEAR** }
- enum [\\_\\_gnu\\_parallel::\\_SortAlgorithm](#) { **MWMS**, **QS**, **QS\_BALANCED** }
- enum [\\_\\_gnu\\_parallel::\\_SplittingAlgorithm](#) { **SAMPLING**, **EXACT** }

### Variables

- static const int [\\_\\_gnu\\_parallel::\\_CASable\\_bits](#)
- static const [\\_CASable](#) [\\_\\_gnu\\_parallel::\\_CASable\\_mask](#)

#### 5.648.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

## 5.649 types\_traits.hpp File Reference

### Classes

- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type](#)< Key, Mapped, \_Alloc, Store\_Hash >
- struct [\\_\\_gnu\\_pbds::detail::maybe\\_null\\_type](#)< Key, null\_type, \_Alloc, Store\_Hash >
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies](#)< Key, Mapped >
- struct [\\_\\_gnu\\_pbds::detail::no\\_throw\\_copies](#)< Key, null\_type >
- struct [\\_\\_gnu\\_pbds::detail::rebind\\_traits](#)< \_Alloc, T >
- struct [\\_\\_gnu\\_pbds::detail::select\\_value\\_type](#)< Key, Mapped >
- struct [\\_\\_gnu\\_pbds::detail::select\\_value\\_type](#)< Key, null\_type >
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data](#)< \_Tv, \_Th, Store\_Hash >
- struct [\\_\\_gnu\\_pbds::detail::stored\\_data](#)< \_Tv, \_Th, false >
- struct [\\_\\_gnu\\_pbds::detail::stored\\_hash](#)< \_Th >
- struct [\\_\\_gnu\\_pbds::detail::stored\\_value](#)< \_Tv >
- struct [\\_\\_gnu\\_pbds::detail::types\\_traits](#)< Key, Mapped, \_Alloc, Store\_Hash >

## Namespaces

- [\\_\\_gnu\\_pbds](#)

### 5.649.1 Detailed Description

Contains a traits class of types used by containers.

## 5.650 `uniform_int_dist.h` File Reference

### Classes

- struct [std::uniform\\_int\\_distribution<\\_IntType>::param\\_type](#)
- class [std::uniform\\_int\\_distribution<\\_IntType>](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Functions

- `template<typename _Tp>  
bool std::\_\_detail::\_Power\_of\_2 (_Tp __x)`

### 5.650.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<random>`.

## 5.651 `unique_copy.h` File Reference

### Namespaces

- [\\_\\_gnu\\_parallel](#)

### Functions

- `template<typename _Iter, class _OutputIterator, class _BinaryPredicate>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result, ↩  
_BinaryPredicate __binary_pred)`
- `template<typename _Iter, class _OutputIterator>  
_OutputIterator \_\_gnu\_parallel::\_\_parallel\_unique\_copy (_Iter __first, _Iter __last, _OutputIterator __result)`

### 5.651.1 Detailed Description

Parallel implementations of `std::unique_copy()`. This file is a GNU parallel extension to the Standard C++ Library.

## 5.652 `unique_lock.h` File Reference

### Classes

- class `std::unique_lock<_Mutex>`

### Namespaces

- `std`

### 5.652.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<mutex>`.

## 5.653 `unique_ptr.h` File Reference

### Classes

- struct `std::default_delete<_Tp>`
- struct `std::default_delete<_Tp[]>`
- struct `std::hash<unique_ptr<_Tp, _Dp>>`
- class `std::unique_ptr<_Tp, _Dp>`
- class `std::unique_ptr<_Tp[], _Dp>`

### Namespaces

- `std`

### Macros

- `#define __cpp_lib_make_unique`

### 5.653.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<memory>`.

## 5.654 unordered\_map File Reference

### Macros

- `#define _GLIBCXX_UNORDERED_MAP`

### 5.654.1 Detailed Description

This is a Standard C++ Library header.

## 5.655 unordered\_map File Reference

### Classes

- class `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc>`

### Namespaces

- `std`
- `std::__debug`

### Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_MAP`

### Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator!= (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator!= (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, const unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_map<_Key, _Tp, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__x, unordered_multimap<_Key, _Tp, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`

## 5.655.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.656 unordered\_map File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_MAP`

## Typedefs

- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **std::experimental::fundamentals\_v2::pmr::unordered\_map** = [std::unordered\\_map](#)<\_Key, \_Tp, \_Hash, \_Pred, polymorphic\_allocator< pair< const \_Key, \_Tp > >>
- `template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **std::experimental::fundamentals\_v2::pmr::unordered\_multimap** = [std::unordered\\_multimap](#)<\_Key, ↵\_Tp, \_Hash, \_Pred, polymorphic\_allocator< pair< const \_Key, \_Tp > >>

## Functions

- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **std::experimental::fundamentals\_v2::erase\_if** (unordered\_map< \_Key, \_Tp, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **std::experimental::fundamentals\_v2::erase\_if** (unordered\_multimap< \_Key, \_Tp, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

## 5.656.1 Detailed Description

This is a TS C++ Library header.

## 5.657 unordered\_map.h File Reference

## Classes

- class [std::unordered\\_map](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multimap](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multimap](#)<\_Key, \_Tp, \_Hash, \_Pred, \_Alloc >

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__umap_hashtable = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::\_\_umap\_traits = __detail::_Hashtable_traits< _Cache, false, true >`
- `template<typename _Key , typename _Tp , typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>  
using std::__ummap_hashtable = _Hashtable< _Key, std::pair< const _Key, _Tp >, _Alloc, __detail::_Select1st, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::\_\_ummap\_traits = __detail::_Hashtable_traits< _Cache, false, false >`

## Functions

- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
bool std::operator!= (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
bool std::operator!= (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
bool std::operator== (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
bool std::operator== (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
void std::swap (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Key , class _Tp , class _Hash , class _Pred , class _Alloc >  
void std::swap (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 5.657.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_map>`.



## 5.658 unordered\_set File Reference

## Macros

- `#define _GLIBCXX_UNORDERED_SET`

## 5.658.1 Detailed Description

This is a Standard C++ Library header.

## 5.659 unordered\_set File Reference

## Classes

- class `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_multiset<_Value, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>`
- class `std::__debug::unordered_set<_Value, _Hash, _Pred, _Alloc>`

## Namespaces

- `std`
- `std::__debug`

## Macros

- `#define _GLIBCXX_DEBUG_UNORDERED_SET`

## Functions

- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`bool std::__debug::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y)`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_set<_Value, _Hash, _Pred, _Alloc> &__x, unordered_set<_Value, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>`  
`void std::__debug::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc> &__y) noexcept(noexcept(__x.swap(__y)))`

### 5.659.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.660 unordered\_set File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define _GLIBCXX_EXPERIMENTAL_UNORDERED_SET`

### Typedefs

- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_multiset`** = [std::unordered\\_multiset](#)< \_Key, \_Hash, \_Pred, polymorphic\_allocator< \_Key > >
- `template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>`  
using **`std::experimental::fundamentals_v2::pmr::unordered_set`** = [std::unordered\\_set](#)< \_Key, \_Hash, \_Pred, polymorphic\_allocator< \_Key > >

### Functions

- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (unordered\_set< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)
- `template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (unordered\_multiset< \_Key, \_Hash, \_CPred, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

### 5.660.1 Detailed Description

This is a TS C++ Library header.

## 5.661 unordered\_set.h File Reference

### Classes

- class [std::unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_multiset](#)< \_Value, \_Hash, \_Pred, \_Alloc >
- class [std::unordered\\_set](#)< \_Value, \_Hash, \_Pred, \_Alloc >

## Namespaces

- [std](#)

## Typedefs

- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__umset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__umset_traits = __detail::_Hashtable_traits<_Cache, true, false >`
- `template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>  
using std::__uset_hashtable = _Hashtable<_Value, _Value, _Alloc, __detail::_Identity, _Pred, _Hash, __detail::_Mod_range_hashing, __detail::_Default_ranged_hash, __detail::_Prime_rehash_policy, _Tr >`
- `template<bool _Cache>  
using std::__uset_traits = __detail::_Hashtable_traits<_Cache, true, true >`

## Functions

- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator!= (const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_set<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
bool std::operator== (const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y)`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_set<_Value, _Hash, _Pred, _Alloc > &__x, unordered_set<_Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`
- `template<class _Value, class _Hash, class _Pred, class _Alloc >  
void std::swap (unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset<_Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))`

## 5.661.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<unordered_set>`.

## 5.662 update\_fn\_imps.hpp File Reference

## 5.662.1 Detailed Description

Contains an implementation class for `pat_trie_`.

## 5.663 utility File Reference

### Classes

- struct [std::\\_\\_is\\_tuple\\_like\\_impl](#)< std::pair< \_T1, \_T2 > >
- struct [std::integer\\_sequence](#)< \_Tp, \_Idx >
- struct [std::tuple\\_element](#)< \_Int, \_Tp >
- struct [std::tuple\\_element](#)< 0, std::pair< \_Tp1, \_Tp2 > >
- struct [std::tuple\\_element](#)< 1, std::pair< \_Tp1, \_Tp2 > >
- struct [std::tuple\\_size](#)< \_Tp >
- struct [std::tuple\\_size](#)< std::pair< \_Tp1, \_Tp2 > >

### Namespaces

- [std](#)

### Macros

- `#define __cpp_lib_exchange_function`
- `#define __cpp_lib_integer_sequence`
- `#define __cpp_lib_tuple_element_t`
- `#define __cpp_lib_tuples_by_type`
- `#define _GLIBCXX_UTILITY`

### Typedefs

- `template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up><::value>::type, size_t = tuple_size<_Tp>::value>  
using std::\_\_enable\_if\_has\_tuple\_size = _Tp`
- `template<std::size_t __i, typename _Tp >  
using std::\_\_tuple\_element\_t = typename tuple_element< __i, _Tp >::type`
- `template<size_t... _Idx>  
using std::index\_sequence = integer_sequence< size_t, _Idx... >`
- `template<typename... _Types>  
using std::index\_sequence\_for = make_index_sequence< sizeof...(_Types)>`
- `template<size_t _Num>  
using std::make\_index\_sequence = make_integer_sequence< size_t, _Num >`
- `template<typename _Tp, _Tp_Num>  
using std::make\_integer\_sequence = integer_sequence< _Tp, __integer_pack(_Num)... >`
- `template<std::size_t __i, typename _Tp >  
using std::tuple\_element\_t = typename tuple_element< __i, _Tp >::type`

## Functions

- `template<typename _Tp, typename _Up = _Tp>  
constexpr _Tp std::exchange (_Tp &__obj, _Up &&__new_val)`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (std::pair< _Tp1, _Tp2 > &__in)  
noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type && std::get (std::pair< _Tp1, _Tp2 > &&__in)  
noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type & std::get (const std::pair< _Tp1, _Tp2 > &__in) noexcept`
- `template<std::size_t _Int, class _Tp1, class _Tp2 >  
constexpr const tuple_element< _Int, std::pair< _Tp1, _Tp2 > >::type && std::get (const std::pair< _Tp1, _Tp2 > &&__in) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp & std::get (pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp & std::get (const pair< _Tp, _Up > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp && std::get (pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp && std::get (const pair< _Tp, _Up > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp & std::get (pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp & std::get (const pair< _Up, _Tp > &__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr _Tp && std::get (pair< _Up, _Tp > &&__p) noexcept`
- `template<typename _Tp, typename _Up >  
constexpr const _Tp && std::get (const pair< _Up, _Tp > &&__p) noexcept`

## 5.663.1 Detailed Description

This is a Standard C++ Library header.

## 5.664 utility File Reference

## Namespaces

- [std](#)
- [std::experimental](#)

## Macros

- `#define \_GLIBCXX\_EXPERIMENTAL\_UTILITY`

## Typedefs

- using **std::experimental::fundamentals\_v2::erased\_type** = std::\_\_erased\_type

### 5.664.1 Detailed Description

This is a TS C++ Library header.

## 5.665 valarray File Reference

### Classes

- class [std::gslice\\_array<\\_Tp>](#)
- class [std::indirect\\_array<\\_Tp>](#)
- class [std::mask\\_array<\\_Tp>](#)
- class [std::slice\\_array<\\_Tp>](#)
- class [std::valarray<\\_Tp>](#)
- class [std::valarray<\\_Tp>](#)

### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

### Macros

- **#define \_DEFINE\_BINARY\_OPERATOR(\_Op, \_Name)**
- **#define \_DEFINE\_VALARRAY\_AUGMENTED\_ASSIGNMENT(\_Op, \_Name)**
- **#define \_DEFINE\_VALARRAY\_EXPR\_AUGMENTED\_ASSIGNMENT(\_Op, \_Name)**
- **#define \_DEFINE\_VALARRAY\_UNARY\_OPERATOR(\_Op, \_Name)**
- **#define \_GLIBCXX\_VALARRAY**

### Functions

- `template<class _Tp>  
_Tp * std::begin (valarray<_Tp> &__va)`
- `template<class _Tp>  
const _Tp * std::begin (const valarray<_Tp> &__va)`
- `template<class _Tp>  
_Tp * std::end (valarray<_Tp> &__va)`
- `template<class _Tp>  
const _Tp * std::end (const valarray<_Tp> &__va)`
- `template<typename _Tp>  
_Expr<_BinClos<__not_equal_to, _ValArray, _Constant, _Tp, _Tp>, typename __fun<__not_equal_to, _Tp>::result_type> > std::operator!= (const valarray<_Tp> &__v, const typename valarray<_Tp>::value_type &__t)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __not_equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __not_equal_to, _Tp >::result_type > std::operator!= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __modulus, _Tp >::result_type > std::operator% (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __modulus, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __modulus, _Tp >::result_type > std::operator% (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp >::result_type > std::operator& (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp >::result_type > std::operator& (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_and, _Tp >::result_type > std::operator& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_and, _Tp >::result_type > std::operator&& (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp >::result_type > std::operator&& (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_and, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_and, _Tp >::result_type > std::operator&& (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __multiplies, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __multiplies, _Tp >::result_type > std::operator* (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __plus, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __plus, _Tp >::result_type > std::operator+ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`

[illegible]



- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > std::operator== (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __equal_to, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __equal_to, _Tp >::result_type > std::operator== (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > std::operator> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater, _Tp >::result_type > std::operator> (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > std::operator>= (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > std::operator>= (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __greater_equal, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __greater_equal, _Tp >::result_type > std::operator>= (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __shift_right, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __shift_right, _Tp >::result_type > std::operator>> (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_xor, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_xor, _Tp >::result_type > std::operator^ (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`

- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > std::operator| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > std::operator| (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __bitwise_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __bitwise_or, _Tp >↵`  
`::result_type > std::operator| (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _ValArray, _Constant, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type`  
`&__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< __logical_or, _Constant, _ValArray, _Tp, _Tp >, typename __fun< __logical_or, _Tp >↵`  
`::result_type > std::operator|| (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &↵`  
`__v)`

#### 5.665.1 Detailed Description

This is a Standard C++ Library header.

#### 5.666 valarray\_after.h File Reference

##### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

##### Macros

- `#define _DEFINE_EXPR_BINARY_FUNCTION(_Fun, _UFun)`
- `#define _DEFINE_EXPR_BINARY_OPERATOR(_Op, _Name)`
- `#define _DEFINE_EXPR_UNARY_FUNCTION(_Name, _UName)`
- `#define _DEFINE_EXPR_UNARY_OPERATOR(_Op, _Name)`

## Functions

- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > std::abs (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > std::abs (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > std::acos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > std::acos (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > std::asin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > std::asin (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > std::atan (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > std::atan (const valarray< _Tp > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std::`  
`atan2 (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _`  
`_Dom2::value_type > &__e2)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray<`  
`typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _`  
`_Dom::value_type > std::atan2 (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, type-`  
`name _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _`  
`_Dom::value_type > std::atan2 (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _`  
`_Dom::value_type > std::atan2 (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _`  
`_Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::atan2 (const valarray<`  
`_Tp > &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::atan2 (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > std::cos (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > std::cos (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > std::cosh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > std::cosh (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > std::exp (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > std::exp (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > std::log (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > std::log (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > std::log10 (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > std::log10 (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_not_equal_to, typename _Dom1::value_type >::result_type > std::operator!= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::_not_equal_to, typename _Dom::value_type >::result_type > std::operator!= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::_modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::_modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_modulus, typename _Dom1::value_type >::result_type > std::operator% (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::_modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::_modulus, typename _Dom::value_type >::result_type > std::operator% (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__modulus, typename _Dom::value_type >::result_type > std::operator% (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > std::operator& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`bitwise_and, typename _Dom1::value_type >::result_type > std::operator& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`logical_and, typename _Dom1::value_type >::result_type > std::operator&& (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name _fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-`  
`name _fun< struct std::__logical_and, typename _Dom::value_type >::result_type > std::operator&& (const`  
`_Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`multiplies, typename _Dom1::value_type >::result_type > std::operator* (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__multiplies, typename _Dom::value_type >::result_type > std::operator* (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type &__v, const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__plus,`  
`typename _Dom1::value_type >::result_type > std::operator+ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type &__v, const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__minus,`  
`typename _Dom1::value_type >::result_type > std::operator- (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__minus, typename _Dom::value_type >::result_type > std::operator- (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type >`  
`__fun< struct std::__divides, typename _Dom1::value_type >::result_type > std::operator/ (const _Expr< _Dom1,`  
`typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > std::operator/ (const typename _Dom,`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > std::operator/ (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`__fun< struct std::__divides, typename _Dom::value_type >::result_type > std::operator/ (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type >`  
`__fun< struct std::__less, typename _Dom1::value_type >::result_type > std::operator< (const _Expr< _Dom1,`  
`typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const typename _Dom,`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type >`  
`__fun< struct std::__less, typename _Dom::value_type >::result_type > std::operator< (const valarray< type-`  
`name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type >`  
`__fun< struct std::__shift_left, typename _Dom1::value_type >::result_type > std::operator<< (const _Expr< _Dom1,`  
`typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type >`  
`__fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__shift_left, typename _Dom::value_type >::result_type > std::operator<< (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`less_equal, typename _Dom1::value_type >::result_type > std::operator<= (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const`  
`typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > std::operator<= (const`  
`valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`equal_to, typename _Dom1::value_type >::result_type > std::operator== (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__equal_to, typename _Dom::value_type >::result_type > std::operator== (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`



- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const _Expr< _Dom,`  
`typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const valarray<`  
`typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`greater, typename _Dom1::value_type >::result_type > std::operator> (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__greater, typename _Dom::value_type >::result_type > std::operator> (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type >`  
`&__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >,`  
`typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-`  
`name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type >`  
`&__e)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`greater_equal, typename _Dom1::value_type >::result_type > std::operator>= (const _Expr< _Dom1, type-`  
`name _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >,`  
`typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > std::operator>=`  
`(const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__`  
`shift_right, typename _Dom1::value_type >::result_type > std::operator>> (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`__fun< struct std::__shift_right, typename _Dom::value_type >::result_type > std::operator>> (const type-`  
`name _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`

- [illegible]

- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_fun< struct std::__logical_or, typename _Dom::value_type >::result_type > std::operator|| (const typename`  
`_Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename _fun< struct std::__`  
`logical_or, typename _Dom1::value_type >::result_type > std::operator|| (const _Expr< _Dom1, typename`  
`_Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom`  
`::value_type > std::pow (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename`  
`_Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename`  
`_Dom::value_type > std::pow (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom`  
`::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::__Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp`  
`> &__v, const typename valarray< _Tp >::value_type &__t)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename`  
`_Dom::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename`  
`_Dom::value_type &__t)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::__Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > std::pow (const valarray< _Tp`  
`> &__v, const valarray< _Tp > &__w)`
- `template<class _Dom >`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom`  
`::value_type > std::pow (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename`  
`_Dom::value_type > &__v)`
- `template<class _Dom1, class _Dom2 >`  
`_Expr< _BinClos< struct std::__Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > std`  
`::pow (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename`  
`_Dom2::value_type > &__e2)`
- `template<typename _Tp >`  
`_Expr< _BinClos< struct std::__Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > std::pow (const typename`  
`valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::__Sin, _Expr, _Dom >, typename _Dom::value_type > std::sin (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::__Sin, _ValArray, _Tp >, _Tp > std::sin (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::__Sinh, _Expr, _Dom >, typename _Dom::value_type > std::sinh (const _Expr<`  
`_Dom, typename _Dom::value_type > &__e)`

- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > std::sinh (const valarray< _Tp > &__v)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > std::sqrt (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > std::sqrt (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > std::tan (const valarray< _Tp > &__v)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > std::tan (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<class _Dom >`  
`_Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > std::tanh (const _Expr< _Dom, typename _Dom::value_type > &__e)`
- `template<typename _Tp >`  
`_Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > std::tanh (const valarray< _Tp > &__v)`

#### 5.666.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.667 `valarray_array.h` File Reference

#### Namespaces

- [std](#)

#### Macros

- `#define _DEFINE_ARRAY_FUNCTION(_Op, _Name)`

#### Functions

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, _Tp *__restrict __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, size_t __s1, _Tp *__restrict __dst, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __b, size_t __n)`

- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __a, size_t __n, _Tp *__restrict __b, const size_t *__restrict __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (const _Tp *__restrict __src, size_t __n, const size_t *__restrict __i, _Tp *__restrict __dst, const size_t *__restrict __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, _Array<_Tp> __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, size_t __s1, _Array<_Tp> __b, size_t __s2)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __a, size_t __n, _Array<_Tp> __b, _Array<size_t> __i)`
- `template<typename _Tp >`  
`void std::__valarray_copy (_Array<_Tp> __src, size_t __n, _Array<size_t> __i, _Array<_Tp> __dst, _Array<size_t> __j)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__b, const _Tp *__e, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, size_t __n, size_t __s, _Tp *__restrict __o)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (const _Tp *__restrict __a, const size_t *__restrict __i, _Tp *__restrict __o, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array<_Tp> __a, _Array<size_t> __i, _Array<_Tp> __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array<_Tp> __a, size_t __n, size_t __s, _Array<_Tp> __b)`
- `template<typename _Tp >`  
`void std::__valarray_default_construct (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_destroy_elements (_Tp *__b, _Tp *__e)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Tp *__restrict __a, const size_t *__restrict __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array<_Tp> __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array<_Tp> __a, size_t __n, size_t __s, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array<_Tp> __a, _Array<size_t> __i, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__valarray_fill_construct (_Tp *__b, _Tp *__e, const _Tp &__t)`

- `template<typename _Tp >`  
`_Tp * std::__valarray_get_storage (size_t)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_max (const _Ta &__a)`
- `template<typename _Ta >`  
`_Ta::value_type std::__valarray_min (const _Ta &__a)`
- `void std::__valarray_release_memory (void *__p)`
- `template<typename _Tp >`  
`_Tp std::__valarray_sum (const _Tp *__f, const _Tp *__l)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_and (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_or (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::__Array_augmented_bitwise_or (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_or (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::__Array_augmented_bitwise_or (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_or (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__bitwise_xor (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __s, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__divides (_Array< _Tp > __a, _Array< bool > __m, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< size_t > __i, const Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`



- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool >`  
`__m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__minus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp`  
`> &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__modulus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`size_t > __i)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__multiplies (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`

- `template<typename _Tp >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___multiplies (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented___plus (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`

- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t >`  
`__i, const _Expr< _Dom, _Tp > &__e, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool`  
`> __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_left (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array<`  
`bool > __m)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b,`  
`size_t __n)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, const _Tp &__t)`
- `template<typename _Tp, class _Dom >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom,`  
`_Tp > &__e, size_t __n)`
- `template<typename _Tp >`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)`

- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp, class _Dom>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)`
- `template<typename _Tp>`  
`void std::Array_augmented__shift_right (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)`

### 5.667.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

## 5.668 valarray\_array.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _VALARRAY_ARRAY_TCC`

### Functions

- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)`
- `template<typename _Tp, class _Dom>`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`
- `template<typename _Tp>`  
`void std::__valarray_copy (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)`

- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)`
- `template<typename _Tp, class _Dom >`  
`void std::__valarray_copy_construct (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)`
- `template<typename _Tp >`  
`void std::__valarray_copy_construct (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)`
- `template<typename _Tp >`  
`void std::__valarray_fill (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)`

#### 5.668.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.669 valarray\_before.h File Reference

#### Namespaces

- [std](#)
- [std::\\_\\_detail](#)

#### 5.669.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<valarray>`.

### 5.670 variant File Reference

#### Macros

- `#define _GLIBCXX_VARIANT`

#### 5.670.1 Detailed Description

This is the `<variant>` C++ Library header.

### 5.671 vector File Reference

#### Macros

- `#define _GLIBCXX_VECTOR`

### 5.671.1 Detailed Description

This is a Standard C++ Library header.

## 5.672 vector File Reference

### Classes

- class [\\_\\_gnu\\_debug::\\_\\_Safe\\_vector<\\_SafeSequence, \\_BaseSequence >](#)
- struct [std::hash<\\_\\_debug::vector<bool, \\_Alloc > >](#)
- class [std::\\_\\_debug::vector<\\_Tp, \\_Allocator >](#)
- class [std::\\_\\_debug::vector<\\_Tp, \\_Allocator >](#)

### Namespaces

- [\\_\\_gnu\\_debug](#)
- [std](#)
- [std::\\_\\_debug](#)

### Macros

- `#define \_GLIBCXX\_DEBUG\_VECTOR`

### Functions

- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator!= (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator< (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator<= (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator== (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator> (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
bool std::\_\_debug::operator>= (const vector<_Tp, _Alloc > &__lhs, const vector<_Tp, _Alloc > &__rhs)`
- `template<typename _Tp, typename _Alloc >  
void std::\_\_debug::swap (vector<_Tp, _Alloc > &__lhs, vector<_Tp, _Alloc > &__rhs) noexcept(*conditional  
*)`

### 5.672.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

## 5.673 vector File Reference

### Namespaces

- [std](#)
- [std::experimental](#)

### Macros

- `#define __cpp_lib_experimental_erase_if`
- `#define _GLIBCXX_EXPERIMENTAL_VECTOR`

### Typedefs

- `template<typename _Tp >`  
using **`std::experimental::fundamentals_v2::pmr::vector`** = [std::vector](#)< \_Tp, `polymorphic_allocator`< \_Tp >  
>

### Functions

- `template<typename _Tp, typename _Alloc, typename _Up >`  
void **`std::experimental::fundamentals_v2::erase`** (`vector`< \_Tp, \_Alloc > &\_\_cont, const \_Up &\_\_value)
- `template<typename _Tp, typename _Alloc, typename _Predicate >`  
void **`std::experimental::fundamentals_v2::erase_if`** (`vector`< \_Tp, \_Alloc > &\_\_cont, \_Predicate \_\_pred)

#### 5.673.1 Detailed Description

This is a TS C++ Library header.

## 5.674 vector.tcc File Reference

### Namespaces

- [std](#)

### Macros

- `#define _VECTOR_TCC`

#### 5.674.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<vector>`.

## 5.675 vstring.h File Reference

### Classes

- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string<\\_CharT, \\_Traits, \\_Alloc, \\_Base>](#)
- struct [std::hash<\\_\\_gnu\\_cxx::\\_\\_u16vstring>](#)
- struct [std::hash<\\_\\_gnu\\_cxx::\\_\\_u32vstring>](#)
- struct [std::hash<\\_\\_gnu\\_cxx::\\_\\_vstring>](#)
- struct [std::hash<\\_\\_gnu\\_cxx::\\_\\_wvstring>](#)

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Functions

- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> basic\\_istream< \\_CharT, \\_Traits > & std::getline \(basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is, \\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_str, \\_CharT \\_\\_delim\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> basic\\_istream< \\_CharT, \\_Traits > & std::getline \(basic\\_istream< \\_CharT, \\_Traits > &\\_\\_is, \\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_str\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_CharT \\*\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> bool \\_\\_gnu\\_cxx::operator!= \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_CharT \\*\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_CharT \\*\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(\\_CharT \\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, const \\_CharT \\*\\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_lhs, \\_CharT \\_\\_rhs\)](#)
- [template<typename \\_CharT, typename \\_Traits, typename \\_Alloc, template< typename, typename, typename > class \\_Base> \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > \\_\\_gnu\\_cxx::operator+ \(\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &&\\_\\_lhs, const \\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base > &\\_\\_rhs\)](#)



- [illegible]

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator<string< \_CharT, \_Traits, \_Alloc, \_Base > &\_\_rhs\)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *  
__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__  
__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_  
string< _CharT, _Traits, _Alloc, _Base > &__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT  
*__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
bool \_\_gnu\_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base >  
&__rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, \_\_gnu\_cxx::\_\_versa\_string<  
_CharT, _Traits, _Alloc, _Base > &__str)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
void \_\_gnu\_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _  
Traits, _Alloc, _Base > &__rhs)`

### 5.675.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

## 5.676 vstring.tcc File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)
- [std](#)

### Macros

- `#define \_VSTRING\_TCC`

## Functions

- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > & __is, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base > & __str, _CharT __delim)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const _CharT * __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, const _CharT * __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
__versa_string< _CharT, _Traits, _Alloc, _Base > \_\_gnu\_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs, _CharT __rhs)`
- `template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>  
basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > & __is, \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base > & __str)`

## 5.676.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.677 vstring\_fwd.h File Reference

## Classes

- class [\\_\\_gnu\\_cxx::\\_\\_rc\\_string\\_base< \\_CharT, \\_Traits, \\_Alloc >](#)
- class [\\_\\_gnu\\_cxx::\\_\\_versa\\_string< \\_CharT, \\_Traits, \\_Alloc, \\_Base >](#)

## Namespaces

- [\\_\\_gnu\\_cxx](#)

## Typedefs

- typedef \_\_versa\_string< char, [std::char\\_traits](#)< char >, [std::allocator](#)< char >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_rc\_string**
- typedef \_\_vstring **\_\_gnu\_cxx::\_\_sso\_string**
- typedef \_\_versa\_string< char16\_t, [std::char\\_traits](#)< char16\_t >, [std::allocator](#)< char16\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_u16rc\_string**
- typedef \_\_u16vstring **\_\_gnu\_cxx::\_\_u16sso\_string**
- typedef \_\_versa\_string< char16\_t > **\_\_gnu\_cxx::\_\_u16vstring**
- typedef \_\_versa\_string< char32\_t, [std::char\\_traits](#)< char32\_t >, [std::allocator](#)< char32\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_u32rc\_string**
- typedef \_\_u32vstring **\_\_gnu\_cxx::\_\_u32sso\_string**
- typedef \_\_versa\_string< char32\_t > **\_\_gnu\_cxx::\_\_u32vstring**
- typedef \_\_versa\_string< char > **\_\_gnu\_cxx::\_\_vstring**
- typedef \_\_versa\_string< wchar\_t, [std::char\\_traits](#)< wchar\_t >, [std::allocator](#)< wchar\_t >, \_\_rc\_string\_base > **\_\_gnu\_cxx::\_\_wrc\_string**
- typedef \_\_wvstring **\_\_gnu\_cxx::\_\_wsso\_string**
- typedef \_\_versa\_string< wchar\_t > **\_\_gnu\_cxx::\_\_wvstring**

### 5.677.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.678 vstring\_util.h File Reference

### Namespaces

- [\\_\\_gnu\\_cxx](#)

### 5.678.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include `<ext/vstring.h>`.

## 5.679 workstealing.h File Reference

### Classes

- struct [\\_\\_gnu\\_parallel::\\_\\_Job](#)< [\\_DifferenceTp](#) >

### Namespaces

- [\\_\\_gnu\\_parallel](#)

## Macros

- `#define _GLIBCXX_JOB_VOLATILE`

## Functions

- `template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result >  
_Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _↵  
_Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >↵  
::difference_type __bound)`

### 5.679.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.

Work stealing is described in

R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. *Journal of the ACM*, 46(5):720-748, 1999.

This file is a GNU parallel extension to the Standard C++ Library.



## Index

- `_AlgorithmStrategy`
  - `__gnu_parallel`, 534
- `_BALLOC_ALIGN_BYTES`
  - `bitmap_allocator.h`, 3992
- `_BinIndex`
  - `__gnu_parallel`, 533
- `_Bit_scan_forward`
  - `__gnu_cxx`, 500
- `_CASable`
  - `__gnu_parallel`, 533
- `_CASable_bits`
  - `__gnu_parallel`, 584
- `_CASable_mask`
  - `__gnu_parallel`, 584
- `_Construct`
  - `std`, 703
- `_DRandomShufflingGlobalData`
  - `__gnu_parallel::DRandomShufflingGlobalData<_RAIter >`, 1061
- `_Destroy`
  - `std`, 703, 704
- `_Destroy_n`
  - `std`, 704
- `_Distance_precision`
  - `__gnu_debug`, 521
- `_FindAlgorithm`
  - `__gnu_parallel`, 534
- `_Find_first`
  - `SGI`, 394
- `_Find_next`
  - `SGI`, 395
- `_GLIBCXX_BAL_QUICKSORT`
  - `features.h`, 4081
- `_GLIBCXX_CALL`
  - `compiletime_settings.h`, 4023
- `_GLIBCXX_DEBUG_VERIFY_COND_AT`
  - `macros.h`, 4154
- `_GLIBCXX_DEQUE_BUF_SIZE`
  - `stl_deque.h`, 4299
- `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`
  - `features.h`, 4081
- `_GLIBCXX_FIND_EQUAL_SPLIT`
  - `features.h`, 4081
- `_GLIBCXX_FIND_GROWING_BLOCKS`
  - `features.h`, 4082
- `_GLIBCXX_HAS_NESTED_TYPE`
  - `type_traits`, 4351
- `_GLIBCXX_MERGESORT`
  - `features.h`, 4082
- `_GLIBCXX_PARALLEL_ASSERTIONS`
  - `compiletime_settings.h`, 4024
- `_GLIBCXX_PARALLEL_CONDITION`
  - `settings.h`, 4261
- `_GLIBCXX_PARALLEL_LENGTH`
  - `multiway_merge.h`, 4170
- `_GLIBCXX_QUICKSORT`
  - `features.h`, 4082
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1`
  - `compiletime_settings.h`, 4024
- `_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB`
  - `compiletime_settings.h`, 4024
- `_GLIBCXX_SCALE_DOWN_FPU`
  - `compiletime_settings.h`, 4024
- `_GLIBCXX_TREE_DYNAMIC_BALANCING`
  - `features.h`, 4082
- `_GLIBCXX_TREE_FULL_COPY`
  - `features.h`, 4083
- `_GLIBCXX_TREE_INITIAL_SPLITTING`
  - `features.h`, 4083
- `_GLIBCXX_VERBOSE_LEVEL`
  - `compiletime_settings.h`, 4025
- `_GLIBCXX_VOLATILE`
  - `partition.h`, 4196
  - `queue.h`, 4210
- `_GuardedIterator`
  - `__gnu_parallel::GuardedIterator<_RAIter, _Compare >`, 1085
- `_LoserTreeBase`
  - `__gnu_parallel::LoserTreeBase<_Tp, _Compare >`, 1151
- `_M_allocate_and_copy`
  - `std::vector<_Tp, _Alloc >`, 3877
- `_M_allocate_single_object`
  - `__gnu_cxx::bitmap_allocator<_Tp >`, 2150
- `_M_attach`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category >`, 1221
  - `__gnu_debug::Safe_iterator_base`, 1232
  - `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence >`, 1239
  - `__gnu_debug::Safe_local_iterator_base`, 1249
- `_M_attach_single`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category >`, 1222
  - `__gnu_debug::Safe_iterator_base`, 1232
  - `__gnu_debug::Safe_local_iterator<_Iterator, _Sequence >`, 1240
  - `__gnu_debug::Safe_local_iterator_base`, 1249
- `_M_attached_to`
  - `__gnu_debug::Safe_iterator<_Iterator, _Sequence, _Category >`, 1222
  - `__gnu_debug::Safe_iterator_base`, 1232

- `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1240
- `__gnu_debug::Safe_local_iterator_base`, 1250
- `_M_before_dereferenceable`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1222
- `_M_begin`
  - `__gnu_parallel::Piece< _DifferenceTp >`, 1186
- `_M_bin_proc`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, 1061
- `_M_bins_begin`
  - `__gnu_parallel::DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1063
- `_M_buf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2505
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3520
  - `std::basic_filebuf< _CharT, _Traits >`, 1408
- `_M_buf_locale`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2505
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3520
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3544
  - `std::basic_filebuf< _CharT, _Traits >`, 1408
  - `std::basic_streambuf< _CharT, _Traits >`, 1908
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2030
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3925
- `_M_buf_size`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2505
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3520
  - `std::basic_filebuf< _CharT, _Traits >`, 1408
- `_M_can_compare`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1223
  - `__gnu_debug::Safe_iterator_base`, 1232
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1240
  - `__gnu_debug::Safe_local_iterator_base`, 1250
- `_M_clear`
  - `__gnu_cxx::free_list`, 2580
- `_M_comp`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, 1153
- `_M_const_iterators`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1216
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1256
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1260
  - `__gnu_debug::Safe_sequence_base`, 1264
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1268
  - `__gnu_debug::Safe_unordered_container_base`, 1272
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 2005
- `std::__debug::deque< _Tp, _Allocator >`, 2440
- `std::__debug::forward_list< _Tp, _Alloc >`, 2549
- `std::__debug::list< _Tp, _Allocator >`, 2830
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 2901
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3020
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 3085
- `std::__debug::set< _Key, _Compare, _Allocator >`, 3451
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3757
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3793
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3829
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3864
- `std::__debug::vector< _Tp, _Allocator >`, 3901
- `_M_const_local_iterators`
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1268
  - `__gnu_debug::Safe_unordered_container_base`, 1272
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3757
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3793
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3829
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3864
- `_M_create_node`
  - `std::list< _Tp, _Alloc >`, 2838
- `_M_create_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2486
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3501
  - `std::basic_filebuf< _CharT, _Traits >`, 1389
- `_M_deallocate_single_object`
  - `__gnu_cxx::bitmap_allocator< _Tp >`, 2150
- `_M_dereferenceable`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1223
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1241
- `_M_destroy_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2487
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3501
  - `std::basic_filebuf< _CharT, _Traits >`, 1390
- `_M_detach`



- `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1223
- `__gnu_debug::Safe_iterator_base`, 1233
- `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1241
- `__gnu_debug::Safe_local_iterator_base`, 1250
- `_M_detach_all`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1215
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1254
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1258
  - `__gnu_debug::Safe_sequence_base`, 1262
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1266
  - `__gnu_debug::Safe_unordered_container_base`, 1270
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1980
  - `std::__debug::deque< _Tp, _Allocator >`, 2439
  - `std::__debug::forward_list< _Tp, _Alloc >`, 2548
  - `std::__debug::list< _Tp, _Allocator >`, 2828
  - `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 2899
  - `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3018
  - `std::__debug::multiset< _Key, _Compare, _Allocator >`, 3083
  - `std::__debug::set< _Key, _Compare, _Allocator >`, 3449
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3756
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3791
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3827
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3862
  - `std::__debug::vector< _Tp, _Allocator >`, 3900
- `_M_detach_single`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1223
  - `__gnu_debug::Safe_iterator_base`, 1233
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1241
  - `__gnu_debug::Safe_local_iterator_base`, 1250
- `_M_detach_singular`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1215
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1254
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1258
- `__gnu_debug::Safe_sequence_base`, 1263
- `__gnu_debug::Safe_unordered_container< _Container >`, 1266
- `__gnu_debug::Safe_unordered_container_base`, 1271
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1980
- `std::__debug::deque< _Tp, _Allocator >`, 2439
- `std::__debug::forward_list< _Tp, _Alloc >`, 2548
- `std::__debug::list< _Tp, _Allocator >`, 2828
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 2899
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3018
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 3083
- `std::__debug::set< _Key, _Compare, _Allocator >`, 3449
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3756
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3791
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3827
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3862
- `std::__debug::vector< _Tp, _Allocator >`, 3900
- `_M_dist`
  - `__gnu_parallel::__DRandomShufflingGlobalData< _RAIter >`, 1061
- `_M_elements_leftover`
  - `__gnu_parallel::__QSBThreadLocal< _RAIter >`, 1199
- `_M_end`
  - `__gnu_parallel::__Piece< _DifferenceTp >`, 1186
- `_M_ext_buf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2505
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3520
  - `std::basic_filebuf< _CharT, _Traits >`, 1408
- `_M_ext_buf_size`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2505
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3520
  - `std::basic_filebuf< _CharT, _Traits >`, 1408
- `_M_ext_next`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2506
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3521
  - `std::basic_filebuf< _CharT, _Traits >`, 1409
- `_M_fill_initialize`
  - `std::deque< _Tp, _Alloc >`, 2414
- `_M_finish_iterator`
  - `__gnu_parallel::__accumulate_selector< _It >`, 864
  - `__gnu_parallel::__adjacent_difference_selector< _It >`, 865
  - `__gnu_parallel::__count_if_selector< _It, _Diff >`,

- [896](#)
- [\\_\\_gnu\\_parallel::\\_\\_count\\_selector< \\_It, \\_Diff >, 898](#)
- [\\_\\_gnu\\_parallel::\\_\\_fill\\_selector< \\_It >, 920](#)
- [\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector< \\_It >, 926](#)
- [\\_\\_gnu\\_parallel::\\_\\_generate\\_selector< \\_It >, 929](#)
- [\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector< \\_It >, 932](#)
- [\\_\\_gnu\\_parallel::\\_\\_identity\\_selector< \\_It >, 934](#)
- [\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >, 936](#)
- [\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector< \\_It, \\_Op, \\_Tp >, 967](#)
- [\\_\\_gnu\\_parallel::\\_\\_replace\\_selector< \\_It, \\_Tp >, 969](#)
- [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector< \\_It >, 971](#)
- [\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector< \\_It >, 973](#)
- [\\_M\\_first](#)
  - [\\_\\_gnu\\_parallel::\\_\\_Job< \\_DifferenceTp >, 1125](#)
- [\\_M\\_first\\_insert](#)
  - [\\_\\_gnu\\_parallel::\\_\\_LoserTreeBase< \\_Tp, \\_Compare >, 1153](#)
- [\\_M\\_gcount](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >, 1472](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >, 1528](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >, 1626](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >, 1678](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >, 1732](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >, 2090](#)
- [\\_M\\_get](#)
  - [\\_\\_gnu\\_cxx::free\\_list, 2580](#)
- [\\_M\\_get\\_mutex](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_forward\\_list< \\_SafeSequence >, 1215](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator< \\_Iterator, \\_Sequence, \\_Category >, 1223](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_iterator\\_base, 1233](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator< \\_Iterator, \\_Sequence >, 1241](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_local\\_iterator\\_base, 1250](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_node\\_sequence< \\_Sequence >, 1254](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence< \\_Sequence >, 1258](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_sequence\\_base, 1263](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container< \\_Container >, 1266](#)
  - [\\_\\_gnu\\_debug::\\_\\_Safe\\_unordered\\_container\\_base, 1271](#)
  - [\\_\\_gnu\\_debug::\\_\\_basic\\_string< \\_CharT, \\_Traits, \\_Allocator >, 1981](#)
  - [std::\\_\\_debug::deque< \\_Tp, \\_Allocator >, 2439](#)
  - [std::\\_\\_debug::forward\\_list< \\_Tp, \\_Alloc >, 2548](#)
  - [std::\\_\\_debug::list< \\_Tp, \\_Allocator >, 2828](#)
  - [std::\\_\\_debug::map< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 2899](#)
  - [std::\\_\\_debug::multimap< \\_Key, \\_Tp, \\_Compare, \\_Allocator >, 3019](#)
  - [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >, 3083](#)
  - [std::\\_\\_debug::set< \\_Key, \\_Compare, \\_Allocator >, 3449](#)
  - [std::\\_\\_debug::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3756](#)
  - [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3792](#)
  - [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3827](#)
  - [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3862](#)
  - [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 3900](#)
- [\\_M\\_get\\_result](#)
  - [std::\\_\\_basic\\_future< \\_Res >, 885](#)
  - [std::future< \\_Res >, 2596](#)
  - [std::future< \\_Res & >, 2599](#)
  - [std::future< void >, 2602](#)
  - [std::shared\\_future< \\_Res >, 3454](#)
  - [std::shared\\_future< \\_Res & >, 3458](#)
  - [std::shared\\_future< void >, 3461](#)
- [\\_M\\_getloc](#)
  - [std::basic\\_fstream< \\_CharT, \\_Traits >, 1424](#)
  - [std::basic\\_ifstream< \\_CharT, \\_Traits >, 1492](#)
  - [std::basic\\_ios< \\_CharT, \\_Traits >, 1546](#)
  - [std::basic\\_iostream< \\_CharT, \\_Traits >, 1579](#)
  - [std::basic\\_istream< \\_CharT, \\_Traits >, 1643](#)
  - [std::basic\\_istreamstream< \\_CharT, \\_Traits, \\_Alloc >, 1696](#)
  - [std::basic\\_ofstream< \\_CharT, \\_Traits >, 1749](#)
  - [std::basic\\_ostream< \\_CharT, \\_Traits >, 1794](#)
  - [std::basic\\_ostreamstream< \\_CharT, \\_Traits, \\_Alloc >, 1838](#)
  - [std::basic\\_stringstream< \\_CharT, \\_Traits, \\_Alloc >, 2043](#)
  - [std::ios\\_base, 2710](#)
- [\\_M\\_global](#)
  - [\\_\\_gnu\\_parallel::\\_\\_QSBThreadLocal< \\_RAIter >, 1199](#)
- [\\_M\\_in\\_beg](#)
  - [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >, 2506](#)
  - [\\_\\_gnu\\_cxx::stdio\\_filebuf< \\_CharT, \\_Traits >, 3521](#)
  - [\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf< \\_CharT, \\_Traits >, 3545](#)
  - [std::basic\\_filebuf< \\_CharT, \\_Traits >, 1409](#)
  - [std::basic\\_streambuf< \\_CharT, \\_Traits >, 1909](#)
  - [std::basic\\_stringbuf< \\_CharT, \\_Traits, \\_Alloc >, 2030](#)
  - [std::wbuffer\\_convert< \\_Codecvt, \\_Elem, \\_Tr >, 3926](#)
- [\\_M\\_in\\_cur](#)
  - [\\_\\_gnu\\_cxx::enc\\_filebuf< \\_CharT >, 2506](#)

- `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3521
- `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3545
- `std::basic_filebuf< _CharT, _Traits >`, 1409
- `std::basic_streambuf< _CharT, _Traits >`, 1909
- `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2030
- `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3926
- `_M_in_end`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2506
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3521
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3545
  - `std::basic_filebuf< _CharT, _Traits >`, 1409
  - `std::basic_streambuf< _CharT, _Traits >`, 1909
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2030
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3926
- `_M_in_same_bucket`
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1241
- `_M_incrementable`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1223
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1241
- `_M_initial`
  - `__gnu_parallel::QSBThreadLocal< _RAIter >`, 1200
- `_M_initialize_map`
  - `std::Deque_base< _Tp, _Alloc >`, 1057
  - `std::deque< _Tp, _Alloc >`, 2415
- `_M_insert`
  - `__gnu_cxx::free_list`, 2580
- `_M_invalidate`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1224
  - `__gnu_debug::Safe_iterator_base`, 1233
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1242
  - `__gnu_debug::Safe_local_iterator_base`, 1250
- `_M_invalidate_all`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1215
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1254
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1258
  - `__gnu_debug::Safe_sequence_base`, 1263
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1266
  - `__gnu_debug::Safe_unordered_container_base`, 1271
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1981
  - `std::debug::deque< _Tp, _Allocator >`, 2439
- `std::debug::forward_list< _Tp, _Alloc >`, 2548
- `std::debug::list< _Tp, _Allocator >`, 2828
- `std::debug::map< _Key, _Tp, _Compare, _Allocator >`, 2899
- `std::debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3019
- `std::debug::multiset< _Key, _Compare, _Allocator >`, 3083
- `std::debug::set< _Key, _Compare, _Allocator >`, 3449
- `std::debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3756
- `std::debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3792
- `std::debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3827
- `std::debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3862
- `std::debug::vector< _Tp, _Allocator >`, 3900
- `_M_invalidate_if`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, 1216
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1255
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1259
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1267
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1981
  - `std::debug::deque< _Tp, _Allocator >`, 2439
  - `std::debug::forward_list< _Tp, _Alloc >`, 2549
  - `std::debug::list< _Tp, _Allocator >`, 2829
  - `std::debug::map< _Key, _Tp, _Compare, _Allocator >`, 2900
  - `std::debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3019
  - `std::debug::multiset< _Key, _Compare, _Allocator >`, 3084
  - `std::debug::set< _Key, _Compare, _Allocator >`, 3450
  - `std::debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3756
  - `std::debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3792
  - `std::debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3828
  - `std::debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3863
  - `std::debug::vector< _Tp, _Allocator >`, 3900
- `_M_invalidate_local_if`
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1267
  - `std::debug::unordered_map< _Key, _Tp, _Hash,`

- `_Pred, _Alloc >`, [3756](#)
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3792](#)
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3828](#)
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3863](#)
- `_M_is_before_begin`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, [1224](#)
- `_M_is_begin`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, [1224](#)
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, [1242](#)
- `_M_is_beginnest`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, [1224](#)
- `_M_is_end`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, [1225](#)
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, [1242](#)
- `_M_iterators`
  - `__gnu_debug::Safe_forward_list< _SafeSequence >`, [1217](#)
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, [1256](#)
  - `__gnu_debug::Safe_sequence< _Sequence >`, [1260](#)
  - `__gnu_debug::Safe_sequence_base`, [1264](#)
  - `__gnu_debug::Safe_unordered_container< _Container >`, [1268](#)
  - `__gnu_debug::Safe_unordered_container_base`, [1272](#)
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, [2006](#)
  - `std::__debug::deque< _Tp, _Allocator >`, [2440](#)
  - `std::__debug::forward_list< _Tp, _Alloc >`, [2550](#)
  - `std::__debug::list< _Tp, _Allocator >`, [2830](#)
  - `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, [2901](#)
  - `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, [3020](#)
  - `std::__debug::multiset< _Key, _Compare, _Allocator >`, [3085](#)
  - `std::__debug::set< _Key, _Compare, _Allocator >`, [3451](#)
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3758](#)
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3794](#)
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3829](#)
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3864](#)
  - `std::__debug::vector< _Tp, _Allocator >`, [3901](#)
- `_M_key`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser`, [1143](#)
- `_M_last`
  - `__gnu_parallel::Job< _DifferenceTp >`, [1125](#)
- `_M_leftover_parts`
  - `__gnu_parallel::QSBThreadLocal< _RAIter >`, [1200](#)
- `_M_load`
  - `__gnu_parallel::Job< _DifferenceTp >`, [1125](#)
- `_M_local_iterators`
  - `__gnu_debug::Safe_unordered_container< _Container >`, [1268](#)
  - `__gnu_debug::Safe_unordered_container_base`, [1272](#)
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3758](#)
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3794](#)
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3829](#)
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3864](#)
- `_M_log_k`
  - `__gnu_parallel::LoserTree< __stable, _Tp, _Compare >`, [1147](#)
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, [1153](#)
- `_M_losers`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >`, [1153](#)
- `_M_mode`
  - `__gnu_cxx::enc_filebuf< _CharT >`, [2506](#)
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, [3521](#)
  - `std::basic_filebuf< _CharT, _Traits >`, [1409](#)
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, [2030](#)
- `_M_new_elements_at_back`
  - `std::deque< _Tp, _Alloc >`, [2415](#)
- `_M_new_elements_at_front`
  - `std::deque< _Tp, _Alloc >`, [2416](#)
- `_M_next`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, [1228](#)
  - `__gnu_debug::Safe_iterator_base`, [1234](#)
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, [1246](#)
  - `__gnu_debug::Safe_local_iterator_base`, [1251](#)
- `_M_num_bins`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAIter >`, [1061](#)
- `_M_num_bits`

- `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >`, 1062
- `_M_num_threads`
  - `__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >`, 1064
  - `__gnu_parallel::_PMWMSSortingData< _RAIter >`, 1189
  - `__gnu_parallel::_QSBThreadLocal< _RAIter >`, 1200
- `_M_offsets`
  - `__gnu_parallel::_PMWMSSortingData< _RAIter >`, 1190
- `_M_out_beg`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2507
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3522
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3545
  - `std::basic_filebuf< _CharT, _Traits >`, 1410
  - `std::basic_streambuf< _CharT, _Traits >`, 1909
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2031
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3926
- `_M_out_cur`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2507
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3522
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3545
  - `std::basic_filebuf< _CharT, _Traits >`, 1410
  - `std::basic_streambuf< _CharT, _Traits >`, 1909
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2031
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3926
- `_M_out_end`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2507
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3522
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3546
  - `std::basic_filebuf< _CharT, _Traits >`, 1410
  - `std::basic_streambuf< _CharT, _Traits >`, 1910
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2031
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3927
- `_M_pback`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2507
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3522
  - `std::basic_filebuf< _CharT, _Traits >`, 1410
- `_M_pback_cur_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2507
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3523
  - `std::basic_filebuf< _CharT, _Traits >`, 1411
- `_M_pback_end_save`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2508
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3523
  - `std::basic_filebuf< _CharT, _Traits >`, 1411
- `_M_pback_init`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2508
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3523
- `std::basic_filebuf< _CharT, _Traits >`, 1411
- `_M_pieces`
  - `__gnu_parallel::_PMWMSSortingData< _RAIter >`, 1190
- `_M_pop_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2416
- `_M_pop_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2416
- `_M_prior`
  - `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >`, 1229
  - `__gnu_debug::_Safe_iterator_base`, 1234
  - `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >`, 1246
  - `__gnu_debug::_Safe_local_iterator_base`, 1252
- `_M_push_back_aux`
  - `std::deque< _Tp, _Alloc >`, 2416
- `_M_push_front_aux`
  - `std::deque< _Tp, _Alloc >`, 2416
- `_M_range_check`
  - `std::deque< _Tp, _Alloc >`, 2417
  - `std::vector< _Tp, _Alloc >`, 3877
- `_M_range_initialize`
  - `std::deque< _Tp, _Alloc >`, 2417, 2418
- `_M_reading`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2508
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3524
  - `std::basic_filebuf< _CharT, _Traits >`, 1412
- `_M_reallocate_map`
  - `std::deque< _Tp, _Alloc >`, 2418
- `_M_reserve_elements_at_back`
  - `std::deque< _Tp, _Alloc >`, 2418
- `_M_reserve_elements_at_front`
  - `std::deque< _Tp, _Alloc >`, 2419
- `_M_reserve_map_at_back`
  - `std::deque< _Tp, _Alloc >`, 2419
- `_M_reserve_map_at_front`
  - `std::deque< _Tp, _Alloc >`, 2419
- `_M_reset`
  - `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >`, 1225
  - `__gnu_debug::_Safe_iterator_base`, 1233
  - `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >`, 1242
  - `__gnu_debug::_Safe_local_iterator_base`, 1251
- `_M_revalidate_singular`
  - `__gnu_debug::_Safe_forward_list< _SafeSequence >`, 1216
  - `__gnu_debug::_Safe_node_sequence< _Sequence >`, 1255
  - `__gnu_debug::_Safe_sequence< _Sequence >`, 1259
  - `__gnu_debug::_Safe_sequence_base`, 1263

- `__gnu_debug::Safe_unordered_container< _Container >`, 1267
- `__gnu_debug::Safe_unordered_container_base`, 1271
- `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1981
- `std::__debug::deque< _Tp, _Allocator >`, 2440
- `std::__debug::forward_list< _Tp, _Alloc >`, 2549
- `std::__debug::list< _Tp, _Allocator >`, 2829
- `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 2900
- `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3019
- `std::__debug::multiset< _Key, _Compare, _Allocator >`, 3084
- `std::__debug::set< _Key, _Compare, _Allocator >`, 3450
- `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3757
- `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3793
- `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3828
- `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3863
- `std::__debug::vector< _Tp, _Allocator >`, 3901
- `_M_samples`
  - `__gnu_parallel::PMWMSortingData< _RAlter >`, 1190
- `_M_sd`
  - `__gnu_parallel::DRSSorterPU< _RAlter, _RandomNumberGenerator >`, 1064
- `_M_seed`
  - `__gnu_parallel::DRSSorterPU< _RAlter, _RandomNumberGenerator >`, 1064
- `_M_sequence`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1229
  - `__gnu_debug::Safe_iterator_base`, 1234
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1246
  - `__gnu_debug::Safe_local_iterator_base`, 1252
- `_M_sequential_algorithm`
  - `__gnu_parallel::adjacent_find_selector`, 867
  - `__gnu_parallel::find_first_of_selector< _Filterator >`, 922
  - `__gnu_parallel::find_if_selector`, 924
  - `__gnu_parallel::mismatch_selector`, 941
- `_M_set_buffer`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2487
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3501
  - `std::basic_filebuf< _CharT, _Traits >`, 1390
- `_M_set_node`
  - `std::Deque_iterator< _Tp, _Ref, _Ptr >`, 1059
- `_M_singular`
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1225
  - `__gnu_debug::Safe_iterator_base`, 1233
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1243
  - `__gnu_debug::Safe_local_iterator_base`, 1251
- `_M_source`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAlter >`, 1062
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser`, 1143
  - `__gnu_parallel::PMWMSortingData< _RAlter >`, 1190
- `_M_starts`
  - `__gnu_parallel::DRandomShufflingGlobalData< _RAlter >`, 1062
  - `__gnu_parallel::PMWMSortingData< _RAlter >`, 1191
- `_M_sup`
  - `__gnu_parallel::LoserTreeBase< _Tp, _Compare >::Loser`, 1144
- `_M_swap`
  - `__gnu_debug::Safe_node_sequence< _Sequence >`, 1255
  - `__gnu_debug::Safe_sequence< _Sequence >`, 1259
  - `__gnu_debug::Safe_sequence_base`, 1263
  - `__gnu_debug::Safe_unordered_container< _Container >`, 1267
  - `__gnu_debug::Safe_unordered_container_base`, 1271, 1272
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1982
  - `std::__debug::deque< _Tp, _Allocator >`, 2440
  - `std::__debug::list< _Tp, _Allocator >`, 2829
  - `std::__debug::map< _Key, _Tp, _Compare, _Allocator >`, 2900
  - `std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3020
  - `std::__debug::multiset< _Key, _Compare, _Allocator >`, 3084
  - `std::__debug::set< _Key, _Compare, _Allocator >`, 3450
  - `std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3757
  - `std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3793
  - `std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3828
  - `std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3863
  - `std::__debug::vector< _Tp, _Allocator >`, 3901
- `_M_temporaries`



- `__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >`, 1062
- `_M_temporary`
  - `__gnu_parallel::_PMWMSortingData< _RAIter >`, 1191
- `_M_transfer_from_if`
  - `__gnu_debug::_Safe_forward_list< _SafeSequence >`, 1216
  - `__gnu_debug::_Safe_node_sequence< _Sequence >`, 1255
  - `__gnu_debug::_Safe_sequence< _Sequence >`, 1259
  - `__gnu_debug::_basic_string< _CharT, _Traits, _Allocator >`, 1982
  - `std::_debug::deque< _Tp, _Allocator >`, 2440
  - `std::_debug::forward_list< _Tp, _Alloc >`, 2549
  - `std::_debug::list< _Tp, _Allocator >`, 2829
  - `std::_debug::map< _Key, _Tp, _Compare, _Allocator >`, 2900
  - `std::_debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3020
  - `std::_debug::multiset< _Key, _Compare, _Allocator >`, 3084
  - `std::_debug::set< _Key, _Compare, _Allocator >`, 3450
  - `std::_debug::vector< _Tp, _Allocator >`, 3901
- `_M_unlink`
  - `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >`, 1225
  - `__gnu_debug::_Safe_iterator_base`, 1234
  - `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >`, 1243
  - `__gnu_debug::_Safe_local_iterator_base`, 1251
- `_M_use_pointer`
  - `__gnu_parallel::_LoserTreeTraits< _Tp >`, 1161
- `_M_version`
  - `__gnu_debug::_Safe_forward_list< _SafeSequence >`, 1217
  - `__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >`, 1229
  - `__gnu_debug::_Safe_iterator_base`, 1235
  - `__gnu_debug::_Safe_local_iterator< _Iterator, _Sequence >`, 1247
  - `__gnu_debug::_Safe_local_iterator_base`, 1252
  - `__gnu_debug::_Safe_node_sequence< _Sequence >`, 1256
  - `__gnu_debug::_Safe_sequence< _Sequence >`, 1260
  - `__gnu_debug::_Safe_sequence_base`, 1264
  - `__gnu_debug::_Safe_unordered_container< _Container >`, 1269
  - `__gnu_debug::_Safe_unordered_container_base`, 1273
  - `__gnu_debug::_basic_string< _CharT, _Traits, _Allocator >`, 2006
- `std::_debug::deque< _Tp, _Allocator >`, 2441
- `std::_debug::forward_list< _Tp, _Alloc >`, 2550
- `std::_debug::list< _Tp, _Allocator >`, 2830
- `std::_debug::map< _Key, _Tp, _Compare, _Allocator >`, 2901
- `std::_debug::multimap< _Key, _Tp, _Compare, _Allocator >`, 3020
- `std::_debug::multiset< _Key, _Compare, _Allocator >`, 3085
- `std::_debug::set< _Key, _Compare, _Allocator >`, 3451
- `std::_debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3758
- `std::_debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3794
- `std::_debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3830
- `std::_debug::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3865
- `std::_debug::vector< _Tp, _Allocator >`, 3902
- `_M_w`
  - `std::_Base_bitset< _Nw >`, 1047
  - `std::tr2::_dynamic_bitset_base< _WordT, _Alloc >`, 919
- `_M_write`
  - `std::basic_fstream< _CharT, _Traits >`, 1424
  - `std::basic_iostream< _CharT, _Traits >`, 1579
  - `std::basic_ofstream< _CharT, _Traits >`, 1750
  - `std::basic_ostream< _CharT, _Traits >`, 1794
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1838
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2044
- `_MultiwayMergeAlgorithm`
  - `__gnu_parallel`, 534
- `_Opcode`
  - Base and Implementation Classes, 31
- `_Parallelism`
  - `__gnu_parallel`, 534
- `_PartialSumAlgorithm`
  - `__gnu_parallel`, 535
- `_Piece`
  - `__gnu_parallel::_QSBThreadLocal< _RAIter >`, 1198
- `_PseudoSequence`
  - `__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >`, 1196
- `_Ptr`
  - `std::_basic_future< _Res >`, 885
  - `std::_future_base`, 928
  - `std::future< _Res >`, 2596
  - `std::future< _Res & >`, 2598
  - `std::future< void >`, 2601

- std::shared\_future< \_Res >, 3453
- std::shared\_future< \_Res & >, 3457
- std::shared\_future< void >, 3460
- \_QSBThreadLocal
  - \_\_gnu\_parallel::\_QSBThreadLocal< \_RAIter >, 1199
- \_RandomNumber
  - \_\_gnu\_parallel::\_RandomNumber, 1201, 1202
- \_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue< \_Tp >, 1206
- \_S\_constant
  - \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 1226
  - \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >, 1243
- \_Safe\_iterator
  - \_\_gnu\_debug::\_Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 1220, 1221
- \_Safe\_iterator\_base
  - \_\_gnu\_debug::\_Safe\_iterator\_base, 1231
- \_Safe\_local\_iterator
  - \_\_gnu\_debug::\_Safe\_local\_iterator< \_Iterator, \_Sequence >, 1237–1239
- \_Safe\_local\_iterator\_base
  - \_\_gnu\_debug::\_Safe\_local\_iterator\_base, 1248, 1249
- \_SequenceIndex
  - \_\_gnu\_parallel, 533
- \_SortAlgorithm
  - \_\_gnu\_parallel, 535
- \_SplittingAlgorithm
  - \_\_gnu\_parallel, 535
- \_Temporary\_buffer
  - std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1292
- \_ThreadIndex
  - \_\_gnu\_parallel, 533
- \_TokenT
  - std::\_\_detail::Scanner< \_CharT >, 1276
- \_Unchecked\_flip
  - SGI, 395
- \_Unchecked\_reset
  - SGI, 396
- \_Unchecked\_set
  - SGI, 396
- \_Unchecked\_test
  - SGI, 396
- \_\_addressof
  - Utilities, 474
- \_\_allocate\_guarded
  - std, 691
- \_\_allocated\_ptr
  - std::\_\_allocated\_ptr< \_Alloc >, 878, 879
- \_\_allocator\_base
  - Allocators, 8
- \_\_base
  - \_\_gnu\_debug, 521
- \_\_begin1\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 936
- \_\_begin2\_iterator
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 936
- \_\_bins\_end
  - \_\_gnu\_parallel::DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 1063
- \_\_bit\_allocate
  - \_\_gnu\_cxx::\_\_detail, 513
- \_\_bit\_free
  - \_\_gnu\_cxx::\_\_detail, 513
- \_\_calc\_borders
  - \_\_gnu\_parallel, 535
- \_\_check\_singular
  - \_\_gnu\_debug, 521
- \_\_check\_singular\_aux
  - \_\_gnu\_debug, 522
- \_\_check\_string
  - \_\_gnu\_debug, 522
- \_\_clp2
  - Base and Implementation Classes, 29
- \_\_compare\_and\_swap
  - \_\_gnu\_parallel, 536
- \_\_cpp\_lib\_make\_unique
  - Pointer Abstractions, 307
- \_\_ctype\_type
  - std::basic\_ios< \_CharT, \_Traits >, 1540
- \_\_cxa\_demangle
  - cxxabi.h, 4050
- \_\_cxxabiv1::\_\_forced\_unwind, 926
- \_\_decode2
  - \_\_gnu\_parallel, 536
- \_\_delete\_min\_insert
  - \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, 1146
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1148
- \_\_determine\_samples
  - \_\_gnu\_parallel, 537
- \_\_encode2
  - \_\_gnu\_parallel, 537
- \_\_equally\_split
  - \_\_gnu\_parallel, 538
- \_\_equally\_split\_point
  - \_\_gnu\_parallel, 538
- \_\_fetch\_and\_add
  - \_\_gnu\_parallel, 539
- \_\_final\_insertion\_sort



- std, 691
- \_\_find\_if
  - std, 692
- \_\_find\_if\_not
  - std, 692
- \_\_find\_if\_not\_n
  - std, 693
- \_\_find\_template
  - \_\_gnu\_parallel, 539–541
- \_\_for\_each\_template\_random\_access
  - \_\_gnu\_parallel, 542
- \_\_for\_each\_template\_random\_access\_ed
  - \_\_gnu\_parallel, 543
- \_\_for\_each\_template\_random\_access\_omp\_loop
  - \_\_gnu\_parallel, 544
- \_\_for\_each\_template\_random\_access\_omp\_loop\_static
  - \_\_gnu\_parallel, 545
- \_\_for\_each\_template\_random\_access\_workstealing
  - \_\_gnu\_parallel, 545
- \_\_foreign\_iterator\_aux2
  - \_\_gnu\_debug, 522, 523
- \_\_from\_chars\_alnum
  - std::\_\_detail, 805
- \_\_from\_chars\_binary
  - std::\_\_detail, 805
- \_\_from\_chars\_digit
  - std::\_\_detail, 805
- \_\_gcd
  - std, 693
- \_\_gen\_two\_uniform\_ints
  - std, 693
- \_\_genrand\_bits
  - \_\_gnu\_parallel::\_\_RandomNumber, 1202
- \_\_get\_distance
  - \_\_gnu\_debug, 523
- \_\_get\_min\_source
  - \_\_gnu\_parallel::\_\_LoserTree< \_\_stable, \_Tp, \_Compare >, 1146
  - \_\_gnu\_parallel::\_\_LoserTree< false, \_Tp, \_Compare >, 1148
  - \_\_gnu\_parallel::\_\_LoserTreeBase< \_Tp, \_Compare >, 1152
- \_\_get\_num\_threads
  - \_\_gnu\_parallel::\_\_balanced\_quicksort\_tag, 1381
  - \_\_gnu\_parallel::\_\_balanced\_tag, 1382
  - \_\_gnu\_parallel::\_\_default\_parallel\_tag, 2401
  - \_\_gnu\_parallel::\_\_exact\_tag, 2521
  - \_\_gnu\_parallel::\_\_multiway\_mergesort\_exact\_tag, 3086
  - \_\_gnu\_parallel::\_\_multiway\_mergesort\_sampling\_tag, 3087
  - \_\_gnu\_parallel::\_\_multiway\_mergesort\_tag, 3089
  - \_\_gnu\_parallel::\_\_omp\_loop\_static\_tag, 3191
  - \_\_gnu\_parallel::\_\_omp\_loop\_tag, 3192
  - \_\_gnu\_parallel::\_\_parallel\_tag, 3235
  - \_\_gnu\_parallel::\_\_quicksort\_tag, 3302
  - \_\_gnu\_parallel::\_\_sampling\_tag, 3404
  - \_\_gnu\_parallel::\_\_unbalanced\_tag, 3693
- \_\_glibcxx\_check\_erase
  - macros.h, 4151
- \_\_glibcxx\_check\_erase\_after
  - macros.h, 4151
- \_\_glibcxx\_check\_erase\_range
  - macros.h, 4151
- \_\_glibcxx\_check\_erase\_range\_after
  - macros.h, 4151
- \_\_glibcxx\_check\_heap\_pred
  - macros.h, 4151
- \_\_glibcxx\_check\_insert
  - macros.h, 4152
- \_\_glibcxx\_check\_insert\_after
  - macros.h, 4152
- \_\_glibcxx\_check\_insert\_range
  - macros.h, 4152
- \_\_glibcxx\_check\_insert\_range\_after
  - macros.h, 4152
- \_\_glibcxx\_check\_partitioned\_lower
  - macros.h, 4153
- \_\_glibcxx\_check\_partitioned\_lower\_pred
  - macros.h, 4153
- \_\_glibcxx\_check\_partitioned\_upper\_pred
  - macros.h, 4153
- \_\_glibcxx\_check\_sorted\_pred
  - macros.h, 4153
- \_\_gnu\_cxx, 484
  - \_Bit\_scan\_forward, 500
  - \_int\_traits, 499
  - \_static\_pointer\_cast, 499
  - operator!=, 500, 501
  - operator<, 504, 505
  - operator<=, 505, 506
  - operator>, 509, 510
  - operator>=, 510, 511
  - operator+, 501–503
  - operator==, 507, 508
  - swap, 512
- \_\_gnu\_cxx::\_\_Caster< \_ToType >, 1052
- \_\_gnu\_cxx::\_\_Char\_types< \_CharT >, 1052
- \_\_gnu\_cxx::\_\_ExtPtr\_allocator< \_Tp >, 1074
- \_\_gnu\_cxx::\_\_Invalid\_type, 1116
- \_\_gnu\_cxx::\_\_Pointer\_adapter< \_Storage\_policy >, 1191
- \_\_gnu\_cxx::\_\_Relative\_pointer\_impl< \_Tp >, 1205
- \_\_gnu\_cxx::\_\_Relative\_pointer\_impl< const \_Tp >, 1205
- \_\_gnu\_cxx::\_\_Std\_pointer\_impl< \_Tp >, 1291
- \_\_gnu\_cxx::\_\_Unqualified\_type< \_Tp >, 1296
- \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 868
  - allocate, 871–873
  - const\_void\_pointer, 870

- construct, [873](#), [874](#)
- deallocate, [874](#), [875](#)
- destroy, [875](#), [876](#)
- is\_always\_equal, [870](#)
- max\_size, [876](#)
- propagate\_on\_container\_copy\_assignment, [870](#)
- propagate\_on\_container\_move\_assignment, [870](#)
- propagate\_on\_container\_swap, [871](#)
- select\_on\_container\_copy\_construction, [877](#)
- void\_pointer, [871](#)
- `__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >`, [894](#)
- `__gnu_cxx::__detail`, [512](#)
  - `__bit_allocate`, [513](#)
  - `__bit_free`, [513](#)
  - `__num_bitmaps`, [513](#)
  - `__num_blocks`, [514](#)
- `__gnu_cxx::__detail::Bitmap_counter< _Tp >`, [1050](#)
- `__gnu_cxx::__detail::Ffit_finder< _Tp >`, [1076](#)
  - `argument_type`, [1076](#)
  - `result_type`, [1077](#)
- `__gnu_cxx::__detail::__mini_vector< _Tp >`, [940](#)
- `__gnu_cxx::__mt_alloc< _Tp, _Poolp >`, [943](#)
- `__gnu_cxx::__mt_alloc_base< _Tp >`, [944](#)
- `__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >`, [954](#)
- `__gnu_cxx::__pool< _Thread >`, [955](#)
- `__gnu_cxx::__pool< false >`, [955](#)
- `__gnu_cxx::__pool< true >`, [956](#)
- `__gnu_cxx::__pool_alloc< _Tp >`, [957](#)
- `__gnu_cxx::__pool_alloc_base`, [959](#)
- `__gnu_cxx::__pool_base`, [960](#)
- `__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >`, [961](#)
- `__gnu_cxx::__scoped_lock`, [970](#)
- `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [975](#)
  - `__versa_string`, [979–982](#)
  - `~__versa_string`, [983](#)
- `append`, [983–986](#)
- `assign`, [987–991](#)
- `at`, [992](#)
- `back`, [993](#)
- `begin`, [993](#), [994](#)
- `c_str`, [994](#)
- `capacity`, [994](#)
- `cbegin`, [994](#)
- `cend`, [995](#)
- `clear`, [995](#)
- `compare`, [995–998](#)
- `copy`, [999](#)
- `crbegin`, [1000](#)
- `crend`, [1000](#)
- `data`, [1000](#)
- `empty`, [1001](#)
- `end`, [1001](#)
- `erase`, [1002](#), [1003](#)
- `find`, [1003–1005](#)
- `find_first_not_of`, [1006–1008](#)
- `find_first_of`, [1008–1010](#)
- `find_last_not_of`, [1010–1012](#)
- `find_last_of`, [1013](#), [1014](#)
- `front`, [1015](#)
- `get_allocator`, [1015](#)
- `insert`, [1016–1021](#)
- `length`, [1022](#)
- `max_size`, [1022](#)
- `npos`, [1044](#)
- `operator+=`, [1022](#), [1024](#), [1025](#)
- `operator=`, [1025–1027](#)
- `operator[]`, [1027](#), [1028](#)
- `pop_back`, [1028](#)
- `push_back`, [1029](#)
- `rbegin`, [1029](#)
- `rend`, [1030](#)
- `replace`, [1030–1038](#)
- `reserve`, [1039](#)
- `resize`, [1039](#), [1040](#)
- `rfind`, [1040–1042](#)
- `shrink_to_fit`, [1042](#)
- `size`, [1043](#)
- `substr`, [1043](#)
- `swap`, [1044](#)
- `__gnu_cxx::annotate_base`, [1321](#)
- `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`, [2117](#)
  - `argument_type`, [2118](#)
  - `result_type`, [2118](#)
- `__gnu_cxx::bitmap_allocator< _Tp >`, [2148](#)
  - `_M_allocate_single_object`, [2150](#)
  - `_M_deallocate_single_object`, [2150](#)
- `__gnu_cxx::char_traits< _CharT >`, [2193](#)
- `__gnu_cxx::character< _Value, _Int, _St >`, [2198](#)
- `__gnu_cxx::condition_base`, [2264](#)
- `__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >`, [2274](#)
- `__gnu_cxx::constant_unary_fun< _Result, _Argument >`, [2276](#)
- `__gnu_cxx::constant_void_fun< _Result >`, [2277](#)
- `__gnu_cxx::debug_allocator< _Alloc >`, [2389](#)
- `__gnu_cxx::enc_filebuf< _CharT >`, [2483](#)
  - `_M_buf`, [2505](#)
  - `_M_buf_locale`, [2505](#)
  - `_M_buf_size`, [2505](#)
  - `_M_create_pback`, [2486](#)
  - `_M_destroy_pback`, [2487](#)
  - `_M_ext_buf`, [2505](#)
  - `_M_ext_buf_size`, [2505](#)

[\\_M\\_ext\\_next](#), 2506  
[\\_M\\_in\\_beg](#), 2506  
[\\_M\\_in\\_cur](#), 2506  
[\\_M\\_in\\_end](#), 2506  
[\\_M\\_mode](#), 2506  
[\\_M\\_out\\_beg](#), 2507  
[\\_M\\_out\\_cur](#), 2507  
[\\_M\\_out\\_end](#), 2507  
[\\_M\\_pback](#), 2507  
[\\_M\\_pback\\_cur\\_save](#), 2507  
[\\_M\\_pback\\_end\\_save](#), 2508  
[\\_M\\_pback\\_init](#), 2508  
[\\_M\\_reading](#), 2508  
[\\_M\\_set\\_buffer](#), 2487  
[close](#), 2487  
[eback](#), 2487  
[egptr](#), 2488  
[epptr](#), 2488  
[gbump](#), 2488  
[getloc](#), 2489  
[gptr](#), 2489  
[imbue](#), 2489  
[in\\_avail](#), 2490  
[is\\_open](#), 2490  
[open](#), 2490, 2491  
[overflow](#), 2492  
[pbackfail](#), 2492  
[pbase](#), 2493  
[pbump](#), 2493  
[pptr](#), 2494  
[pubimbue](#), 2494  
[pubseekoff](#), 2494  
[pubseekpos](#), 2495  
[pubsetbuf](#), 2495  
[pubsync](#), 2496  
[sbumpc](#), 2496  
[seekoff](#), 2496  
[seekpos](#), 2496  
[setbuf](#), 2497  
[setg](#), 2497  
[setp](#), 2498  
[sgetc](#), 2498  
[sgetn](#), 2499  
[showmanyc](#), 2499  
[snextc](#), 2500  
[sputbackc](#), 2500  
[sputc](#), 2501  
[sputn](#), 2501  
[sungetc](#), 2502  
[sync](#), 2502  
[uflow](#), 2502  
[underflow](#), 2503  
[xsgetn](#), 2503  
[xsputn](#), 2504  
  
[\\_\\_gnu\\_cxx::encoding\\_char\\_traits<\\_CharT>](#), 2509  
[\\_\\_gnu\\_cxx::encoding\\_state](#), 2510  
[\\_\\_gnu\\_cxx::forced\\_error](#), 2544  
[\\_\\_gnu\\_cxx::free\\_list](#), 2579  
    [\\_M\\_clear](#), 2580  
    [\\_M\\_get](#), 2580  
    [\\_M\\_insert](#), 2580  
[\\_\\_gnu\\_cxx::hash\\_map<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#), 2669  
[\\_\\_gnu\\_cxx::hash\\_multimap<\\_Key, \\_Tp, \\_HashFn, \\_EqualKey, \\_Alloc>](#), 2671  
[\\_\\_gnu\\_cxx::hash\\_multiset<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#), 2673  
[\\_\\_gnu\\_cxx::hash\\_set<\\_Value, \\_HashFcn, \\_EqualKey, \\_Alloc>](#), 2676  
[\\_\\_gnu\\_cxx::limit\\_condition](#), 2816  
[\\_\\_gnu\\_cxx::limit\\_condition::always\\_adjustor](#), 1320  
[\\_\\_gnu\\_cxx::limit\\_condition::limit\\_adjustor](#), 2816  
[\\_\\_gnu\\_cxx::limit\\_condition::never\\_adjustor](#), 3098  
[\\_\\_gnu\\_cxx::malloc\\_allocator<\\_Tp>](#), 2895  
[\\_\\_gnu\\_cxx::new\\_allocator<\\_Tp>](#), 3098  
[\\_\\_gnu\\_cxx::project1st<\\_Arg1, \\_Arg2>](#), 3289  
    [first\\_argument\\_type](#), 3290  
    [result\\_type](#), 3290  
    [second\\_argument\\_type](#), 3290  
[\\_\\_gnu\\_cxx::project2nd<\\_Arg1, \\_Arg2>](#), 3290  
    [first\\_argument\\_type](#), 3291  
    [result\\_type](#), 3291  
    [second\\_argument\\_type](#), 3291  
[\\_\\_gnu\\_cxx::random\\_condition](#), 3304  
[\\_\\_gnu\\_cxx::random\\_condition::always\\_adjustor](#), 1320  
[\\_\\_gnu\\_cxx::random\\_condition::group\\_adjustor](#), 2631  
[\\_\\_gnu\\_cxx::random\\_condition::never\\_adjustor](#), 3098  
[\\_\\_gnu\\_cxx::rb\\_tree<\\_Key, \\_Value, \\_KeyOfValue, \\_Compare, \\_Alloc>](#), 3323  
[\\_\\_gnu\\_cxx::recursive\\_init\\_error](#), 3338  
[\\_\\_gnu\\_cxx::rope<\\_CharT, \\_Alloc>](#), 3376  
[\\_\\_gnu\\_cxx::select1st<\\_Pair>](#), 3408  
    [argument\\_type](#), 3409  
    [result\\_type](#), 3409  
[\\_\\_gnu\\_cxx::select2nd<\\_Pair>](#), 3409  
    [argument\\_type](#), 3410  
    [result\\_type](#), 3410  
[\\_\\_gnu\\_cxx::slist<\\_Tp, \\_Alloc>](#), 3484  
[\\_\\_gnu\\_cxx::stdio\\_filebuf<\\_CharT, \\_Traits>](#), 3496  
    [\\_M\\_buf](#), 3520  
    [\\_M\\_buf\\_locale](#), 3520  
    [\\_M\\_buf\\_size](#), 3520  
    [\\_M\\_create\\_pback](#), 3501  
    [\\_M\\_destroy\\_pback](#), 3501  
    [\\_M\\_ext\\_buf](#), 3520  
    [\\_M\\_ext\\_buf\\_size](#), 3520  
    [\\_M\\_ext\\_next](#), 3521  
    [\\_M\\_in\\_beg](#), 3521

[\\_M\\_in\\_cur](#), [3521](#)  
[\\_M\\_in\\_end](#), [3521](#)  
[\\_M\\_mode](#), [3521](#)  
[\\_M\\_out\\_beg](#), [3522](#)  
[\\_M\\_out\\_cur](#), [3522](#)  
[\\_M\\_out\\_end](#), [3522](#)  
[\\_M\\_pback](#), [3522](#)  
[\\_M\\_pback\\_cur\\_save](#), [3523](#)  
[\\_M\\_pback\\_end\\_save](#), [3523](#)  
[\\_M\\_pback\\_init](#), [3523](#)  
[\\_M\\_reading](#), [3524](#)  
[\\_M\\_set\\_buffer](#), [3501](#)  
[~stdio\\_filebuf](#), [3501](#)  
[close](#), [3501](#)  
[eback](#), [3502](#)  
[egptr](#), [3502](#)  
[epptr](#), [3502](#)  
[fd](#), [3503](#)  
[file](#), [3503](#)  
[gbump](#), [3503](#)  
[getloc](#), [3504](#)  
[gptr](#), [3504](#)  
[imbue](#), [3504](#)  
[in\\_avail](#), [3505](#)  
[is\\_open](#), [3505](#)  
[open](#), [3505](#), [3506](#)  
[overflow](#), [3507](#)  
[pbackfail](#), [3507](#)  
[pbase](#), [3508](#)  
[pbump](#), [3508](#)  
[pptr](#), [3509](#)  
[pubimbue](#), [3509](#)  
[pubseekoff](#), [3509](#)  
[pubseekpos](#), [3510](#)  
[pubsetbuf](#), [3510](#)  
[pubsync](#), [3511](#)  
[sbumpc](#), [3511](#)  
[seekoff](#), [3511](#)  
[seekpos](#), [3511](#)  
[setbuf](#), [3512](#)  
[setg](#), [3512](#)  
[setp](#), [3513](#)  
[sgetc](#), [3513](#)  
[sgetn](#), [3514](#)  
[showmanyc](#), [3514](#)  
[snextc](#), [3515](#)  
[sputbackc](#), [3515](#)  
[sputc](#), [3516](#)  
[sputn](#), [3516](#)  
[stdio\\_filebuf](#), [3499](#), [3500](#)  
[sungetc](#), [3517](#)  
[sync](#), [3517](#)  
[uflow](#), [3517](#)  
[underflow](#), [3518](#)  
  
[xsgetn](#), [3518](#)  
[xspn](#), [3519](#)  
[\\_\\_gnu\\_cxx::stdio\\_sync\\_filebuf<\\_CharT, \\_Traits>](#), [3524](#)  
[\\_M\\_buf\\_locale](#), [3544](#)  
[\\_M\\_in\\_beg](#), [3545](#)  
[\\_M\\_in\\_cur](#), [3545](#)  
[\\_M\\_in\\_end](#), [3545](#)  
[\\_M\\_out\\_beg](#), [3545](#)  
[\\_M\\_out\\_cur](#), [3545](#)  
[\\_M\\_out\\_end](#), [3546](#)  
[eback](#), [3527](#)  
[egptr](#), [3527](#)  
[epptr](#), [3528](#)  
[file](#), [3528](#)  
[gbump](#), [3528](#)  
[getloc](#), [3529](#)  
[gptr](#), [3529](#)  
[imbue](#), [3529](#)  
[in\\_avail](#), [3530](#)  
[overflow](#), [3530](#)  
[pbackfail](#), [3531](#)  
[pbase](#), [3531](#)  
[pbump](#), [3532](#)  
[pptr](#), [3532](#)  
[pubimbue](#), [3532](#)  
[pubseekoff](#), [3533](#)  
[pubseekpos](#), [3533](#)  
[pubsetbuf](#), [3535](#)  
[pubsync](#), [3535](#)  
[sbumpc](#), [3535](#)  
[seekoff](#), [3535](#)  
[seekpos](#), [3536](#)  
[setbuf](#), [3536](#)  
[setg](#), [3537](#)  
[setp](#), [3537](#)  
[sgetc](#), [3538](#)  
[sgetn](#), [3538](#)  
[showmanyc](#), [3539](#)  
[snextc](#), [3539](#)  
[sputbackc](#), [3540](#)  
[sputc](#), [3540](#)  
[sputn](#), [3541](#)  
[sungetc](#), [3541](#)  
[sync](#), [3542](#)  
[uflow](#), [3542](#)  
[underflow](#), [3542](#)  
[xsgetn](#), [3543](#)  
[xspn](#), [3544](#)  
[\\_\\_gnu\\_cxx::subtractive\\_rng](#), [3572](#)  
[argument\\_type](#), [3573](#)  
[operator\(\)](#), [3573](#)  
[result\\_type](#), [3573](#)  
[subtractive\\_rng](#), [3573](#)

- `__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >`, 3577
  - `~temporary_buffer`, 3578
  - `begin`, 3579
  - `end`, 3579
  - `requested_size`, 3579
  - `size`, 3579
  - `temporary_buffer`, 3578
- `__gnu_cxx::throw_allocator_base< _Tp, _Cond >`, 3584
- `__gnu_cxx::throw_allocator_limit< _Tp >`, 3586
- `__gnu_cxx::throw_allocator_random< _Tp >`, 3588
- `__gnu_cxx::throw_value_base< _Cond >`, 3589
- `__gnu_cxx::throw_value_limit`, 3591
- `__gnu_cxx::throw_value_random`, 3592
- `__gnu_cxx::typelist`, 514
  - `apply_generator`, 515
- `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`, 3688
  - `argument_type`, 3689
  - `result_type`, 3689
- `__gnu_debug`, 515
  - `_Distance_precision`, 521
  - `_base`, 521
  - `_check_singular`, 521
  - `_check_singular_aux`, 522
  - `_check_string`, 522
  - `_foreign_iterator_aux2`, 522, 523
  - `_get_distance`, 523
  - `_valid_range`, 523, 524
  - `_valid_range_aux`, 524
- `__gnu_debug:: _After_nth_from< _Iterator >`, 1045
- `__gnu_debug:: _BeforeBeginHelper< _Sequence >`, 1049
- `__gnu_debug:: _Equal_to< _Type >`, 1067
- `__gnu_debug:: _Not_equal_to< _Type >`, 1184
- `__gnu_debug:: _Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >`, 1213
- `__gnu_debug:: _Safe_forward_list< _SafeSequence >`, 1214
  - `_M_const_iterators`, 1216
  - `_M_detach_all`, 1215
  - `_M_detach_singular`, 1215
  - `_M_get_mutex`, 1215
  - `_M_invalidate_all`, 1215
  - `_M_invalidate_if`, 1216
  - `_M_iterators`, 1217
  - `_M_revalidate_singular`, 1216
  - `_M_transfer_from_if`, 1216
  - `_M_version`, 1217
- `__gnu_debug:: _Safe_iterator< _Iterator, _Sequence, _Category >`, 1217
  - `_M_attach`, 1221
  - `_M_attach_single`, 1222
  - `_M_attached_to`, 1222
  - `_M_before_dereferenceable`, 1222
  - `_M_can_compare`, 1223
  - `_M_dereferenceable`, 1223
  - `_M_detach`, 1223
  - `_M_detach_single`, 1223
  - `_M_get_mutex`, 1223
  - `_M_incrementable`, 1223
  - `_M_invalidate`, 1224
  - `_M_is_before_begin`, 1224
  - `_M_is_begin`, 1224
  - `_M_is_beginnest`, 1224
  - `_M_is_end`, 1225
  - `_M_next`, 1228
  - `_M_prior`, 1229
  - `_M_reset`, 1225
  - `_M_sequence`, 1229
  - `_M_singular`, 1225
  - `_M_unlink`, 1225
  - `_M_version`, 1229
  - `_S_constant`, 1226
  - `_Safe_iterator`, 1220, 1221
  - `base`, 1226
  - `operator _Iterator`, 1226
  - `operator*`, 1226
  - `operator++`, 1227
  - `operator->`, 1227
  - `operator=`, 1228
- `__gnu_debug:: _Safe_iterator_base`, 1230
  - `_M_attach`, 1232
  - `_M_attach_single`, 1232
  - `_M_attached_to`, 1232
  - `_M_can_compare`, 1232
  - `_M_detach`, 1233
  - `_M_detach_single`, 1233
  - `_M_get_mutex`, 1233
  - `_M_invalidate`, 1233
  - `_M_next`, 1234
  - `_M_prior`, 1234
  - `_M_reset`, 1233
  - `_M_sequence`, 1234
  - `_M_singular`, 1233
  - `_M_unlink`, 1234
  - `_M_version`, 1235
  - `_Safe_iterator_base`, 1231
- `__gnu_debug:: _Safe_local_iterator< _Iterator, _Sequence >`, 1235
  - `_M_attach`, 1239
  - `_M_attach_single`, 1240
  - `_M_attached_to`, 1240
  - `_M_can_compare`, 1240
  - `_M_dereferenceable`, 1241
  - `_M_detach`, 1241
  - `_M_detach_single`, 1241
  - `_M_get_mutex`, 1241
  - `_M_in_same_bucket`, 1241

- [\\_M\\_incrementable, 1241](#)
- [\\_M\\_invalidate, 1242](#)
- [\\_M\\_is\\_begin, 1242](#)
- [\\_M\\_is\\_end, 1242](#)
- [\\_M\\_next, 1246](#)
- [\\_M\\_prior, 1246](#)
- [\\_M\\_reset, 1242](#)
- [\\_M\\_sequence, 1246](#)
- [\\_M\\_singular, 1243](#)
- [\\_M\\_unlink, 1243](#)
- [\\_M\\_version, 1247](#)
- [\\_S\\_constant, 1243](#)
- [\\_Safe\\_local\\_iterator, 1237–1239](#)
- [base, 1243](#)
- [bucket, 1244](#)
- [operator \\_iterator, 1244](#)
- [operator\\*, 1244](#)
- [operator++, 1244, 1245](#)
- [operator->, 1245](#)
- [operator=, 1245, 1246](#)
- [\\_\\_gnu\\_debug:: Safe\\_local\\_iterator\\_base, 1247](#)
  - [\\_M\\_attach, 1249](#)
  - [\\_M\\_attach\\_single, 1249](#)
  - [\\_M\\_attached\\_to, 1250](#)
  - [\\_M\\_can\\_compare, 1250](#)
  - [\\_M\\_detach, 1250](#)
  - [\\_M\\_detach\\_single, 1250](#)
  - [\\_M\\_get\\_mutex, 1250](#)
  - [\\_M\\_invalidate, 1250](#)
  - [\\_M\\_next, 1251](#)
  - [\\_M\\_prior, 1252](#)
  - [\\_M\\_reset, 1251](#)
  - [\\_M\\_sequence, 1252](#)
  - [\\_M\\_singular, 1251](#)
  - [\\_M\\_unlink, 1251](#)
  - [\\_M\\_version, 1252](#)
  - [\\_Safe\\_local\\_iterator\\_base, 1248, 1249](#)
- [\\_\\_gnu\\_debug:: Safe\\_node\\_sequence< \\_Sequence >, 1253](#)
  - [\\_M\\_const\\_iterators, 1256](#)
  - [\\_M\\_detach\\_all, 1254](#)
  - [\\_M\\_detach\\_singular, 1254](#)
  - [\\_M\\_get\\_mutex, 1254](#)
  - [\\_M\\_invalidate\\_all, 1254](#)
  - [\\_M\\_invalidate\\_if, 1255](#)
  - [\\_M\\_iterators, 1256](#)
  - [\\_M\\_revalidate\\_singular, 1255](#)
  - [\\_M\\_swap, 1255](#)
  - [\\_M\\_transfer\\_from\\_if, 1255](#)
  - [\\_M\\_version, 1256](#)
- [\\_\\_gnu\\_debug:: Safe\\_sequence< \\_Sequence >, 1257](#)
  - [\\_M\\_const\\_iterators, 1260](#)
  - [\\_M\\_detach\\_all, 1258](#)
  - [\\_M\\_detach\\_singular, 1258](#)
  - [\\_M\\_get\\_mutex, 1258](#)
  - [\\_M\\_invalidate\\_all, 1258](#)
  - [\\_M\\_invalidate\\_if, 1259](#)
  - [\\_M\\_iterators, 1260](#)
  - [\\_M\\_revalidate\\_singular, 1259](#)
  - [\\_M\\_swap, 1259](#)
  - [\\_M\\_transfer\\_from\\_if, 1259](#)
  - [\\_M\\_version, 1260](#)
- [\\_\\_gnu\\_debug:: Safe\\_sequence\\_base, 1261](#)
  - [\\_M\\_const\\_iterators, 1264](#)
  - [\\_M\\_detach\\_all, 1262](#)
  - [\\_M\\_detach\\_singular, 1263](#)
  - [\\_M\\_get\\_mutex, 1263](#)
  - [\\_M\\_invalidate\\_all, 1263](#)
  - [\\_M\\_iterators, 1264](#)
  - [\\_M\\_revalidate\\_singular, 1263](#)
  - [\\_M\\_swap, 1263](#)
  - [\\_M\\_version, 1264](#)
  - [~\\_Safe\\_sequence\\_base, 1262](#)
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container< \\_Container >, 1265](#)
  - [\\_M\\_const\\_iterators, 1268](#)
  - [\\_M\\_const\\_local\\_iterators, 1268](#)
  - [\\_M\\_detach\\_all, 1266](#)
  - [\\_M\\_detach\\_singular, 1266](#)
  - [\\_M\\_get\\_mutex, 1266](#)
  - [\\_M\\_invalidate\\_all, 1266](#)
  - [\\_M\\_invalidate\\_if, 1267](#)
  - [\\_M\\_invalidate\\_local\\_if, 1267](#)
  - [\\_M\\_iterators, 1268](#)
  - [\\_M\\_local\\_iterators, 1268](#)
  - [\\_M\\_revalidate\\_singular, 1267](#)
  - [\\_M\\_swap, 1267](#)
  - [\\_M\\_version, 1269](#)
- [\\_\\_gnu\\_debug:: Safe\\_unordered\\_container\\_base, 1269](#)
  - [\\_M\\_const\\_iterators, 1272](#)
  - [\\_M\\_const\\_local\\_iterators, 1272](#)
  - [\\_M\\_detach\\_all, 1270](#)
  - [\\_M\\_detach\\_singular, 1271](#)
  - [\\_M\\_get\\_mutex, 1271](#)
  - [\\_M\\_invalidate\\_all, 1271](#)
  - [\\_M\\_iterators, 1272](#)
  - [\\_M\\_local\\_iterators, 1272](#)
  - [\\_M\\_revalidate\\_singular, 1271](#)
  - [\\_M\\_swap, 1271, 1272](#)
  - [\\_M\\_version, 1273](#)
  - [~\\_Safe\\_unordered\\_container\\_base, 1270](#)
- [\\_\\_gnu\\_debug:: Safe\\_vector< \\_SafeSequence, \\_BaseSequence >, 1273](#)
- [\\_\\_gnu\\_debug:: Sequence\\_traits< \\_Sequence >, 1276](#)
- [\\_\\_gnu\\_debug:: basic\\_string< \\_CharT, \\_Traits, \\_Allocator >, 1975](#)
  - [\\_M\\_const\\_iterators, 2005](#)
  - [\\_M\\_detach\\_all, 1980](#)

- [\\_M\\_detach\\_singular](#), 1980
- [\\_M\\_get\\_mutex](#), 1981
- [\\_M\\_invalidate\\_all](#), 1981
- [\\_M\\_invalidate\\_if](#), 1981
- [\\_M\\_iterators](#), 2006
- [\\_M\\_revalidate\\_singular](#), 1981
- [\\_M\\_swap](#), 1982
- [\\_M\\_transfer\\_from\\_if](#), 1982
- [\\_M\\_version](#), 2006
- [append](#), 1982, 1983
- [assign](#), 1984
- [at](#), 1985, 1986
- [back](#), 1986
- [capacity](#), 1987
- [compare](#), 1987, 1989
- [empty](#), 1990
- [erase](#), 1990
- [find](#), 1991
- [find\\_first\\_not\\_of](#), 1991
- [find\\_first\\_of](#), 1992
- [find\\_last\\_not\\_of](#), 1992
- [find\\_last\\_of](#), 1993
- [front](#), 1993
- [get\\_allocator](#), 1994
- [insert](#), 1994–1997
- [length](#), 1997
- [max\\_size](#), 1998
- [npos](#), 2006
- [operator+=](#), 1998
- [replace](#), 1998–2003
- [reserve](#), 2004
- [rfind](#), 2004
- [size](#), 2005
- [swap](#), 2005
- [\\_\\_gnu\\_internal](#), 525
- [\\_\\_gnu\\_parallel](#), 525
  - [\\_AlgorithmStrategy](#), 534
  - [\\_BinIndex](#), 533
  - [\\_CASable](#), 533
  - [\\_CASable\\_bits](#), 584
  - [\\_CASable\\_mask](#), 584
  - [\\_FindAlgorithm](#), 534
  - [\\_MultiwayMergeAlgorithm](#), 534
  - [\\_Parallelism](#), 534
  - [\\_PartialSumAlgorithm](#), 535
  - [\\_SequenceIndex](#), 533
  - [\\_SortAlgorithm](#), 535
  - [\\_SplittingAlgorithm](#), 535
  - [\\_ThreadIndex](#), 533
  - [\\_\\_calc\\_borders](#), 535
  - [\\_\\_compare\\_and\\_swap](#), 536
  - [\\_\\_decode2](#), 536
  - [\\_\\_determine\\_samples](#), 537
  - [\\_\\_encode2](#), 537
  - [\\_\\_equally\\_split](#), 538
  - [\\_\\_equally\\_split\\_point](#), 538
  - [\\_\\_fetch\\_and\\_add](#), 539
  - [\\_\\_find\\_template](#), 539–541
  - [\\_\\_for\\_each\\_template\\_random\\_access](#), 542
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_ed](#), 543
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop](#), 544
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_omp\\_loop\\_static](#), 545
  - [\\_\\_for\\_each\\_template\\_random\\_access\\_workstealing](#), 545
  - [\\_\\_is\\_sorted](#), 546
  - [\\_\\_median\\_of\\_three\\_iterators](#), 547
  - [\\_\\_merge\\_advance](#), 547
  - [\\_\\_merge\\_advance\\_movc](#), 548
  - [\\_\\_merge\\_advance\\_usual](#), 549
  - [\\_\\_parallel\\_merge\\_advance](#), 549, 550
  - [\\_\\_parallel\\_nth\\_element](#), 551
  - [\\_\\_parallel\\_partial\\_sort](#), 551
  - [\\_\\_parallel\\_partial\\_sum](#), 552
  - [\\_\\_parallel\\_partial\\_sum\\_basecase](#), 552
  - [\\_\\_parallel\\_partial\\_sum\\_linear](#), 553
  - [\\_\\_parallel\\_partition](#), 554
  - [\\_\\_parallel\\_random\\_shuffle](#), 554
  - [\\_\\_parallel\\_random\\_shuffle\\_drs](#), 555
  - [\\_\\_parallel\\_random\\_shuffle\\_drs\\_pu](#), 555
  - [\\_\\_parallel\\_sort](#), 556–560
  - [\\_\\_parallel\\_sort\\_qs](#), 561
  - [\\_\\_parallel\\_sort\\_qs\\_conquer](#), 562
  - [\\_\\_parallel\\_sort\\_qs\\_divide](#), 562
  - [\\_\\_parallel\\_sort\\_qsb](#), 563
  - [\\_\\_parallel\\_unique\\_copy](#), 563, 564
  - [\\_\\_qsb\\_conquer](#), 564
  - [\\_\\_qsb\\_divide](#), 565
  - [\\_\\_qsb\\_local\\_sort\\_with\\_helping](#), 566
  - [\\_\\_random\\_number\\_pow2](#), 566
  - [\\_\\_rd\\_log2](#), 567
  - [\\_\\_round\\_up\\_to\\_pow2](#), 567
  - [\\_\\_search\\_template](#), 567
  - [\\_\\_sequential\\_multiway\\_merge](#), 568
  - [\\_\\_sequential\\_random\\_shuffle](#), 569
  - [\\_\\_shrink](#), 569
  - [\\_\\_shrink\\_and\\_double](#), 570
  - [\\_\\_yield](#), 570
- [list\\_partition](#), 570
- [max](#), 571
- [min](#), 571
- [multiseq\\_partition](#), 572
- [multiseq\\_selection](#), 572
- [multiway\\_merge](#), 573
- [multiway\\_merge\\_3\\_variant](#), 575
- [multiway\\_merge\\_4\\_variant](#), 576
- [multiway\\_merge\\_exact\\_splitting](#), 576



- multiway\_merge\_loser\_tree, 577
- multiway\_merge\_loser\_tree\_sentinel, 578
- multiway\_merge\_loser\_tree\_unguarded, 578
- multiway\_merge\_sampling\_splitting, 579
- multiway\_merge\_sentinels, 580
- parallel\_balanced, 534
- parallel\_multiway\_merge, 582
- parallel\_omp\_loop, 534
- parallel\_omp\_loop\_static, 534
- parallel\_sort\_mwms, 582
- parallel\_sort\_mwms\_pu, 583
- parallel\_taskqueue, 534
- parallel\_unbalanced, 534
- sequential, 534
- \_\_gnu\_parallel::DRSSorterPU< \_RAIter, \_RandomNumberGenerator >, 1063
  - \_M\_bins\_begin, 1063
  - \_M\_num\_threads, 1064
  - \_M\_sd, 1064
  - \_M\_seed, 1064
  - \_bins\_end, 1063
- \_\_gnu\_parallel::DRandomShufflingGlobalData< \_RAIter >, 1060
  - \_DRandomShufflingGlobalData, 1061
  - \_M\_bin\_proc, 1061
  - \_M\_dist, 1061
  - \_M\_num\_bins, 1061
  - \_M\_num\_bits, 1062
  - \_M\_source, 1062
  - \_M\_starts, 1062
  - \_M\_temporaries, 1062
- \_\_gnu\_parallel::DummyReduct, 1065
- \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare >, 1068
  - first\_argument\_type, 1069
  - result\_type, 1069
  - second\_argument\_type, 1069
- \_\_gnu\_parallel::EqualTo< \_T1, \_T2 >, 1072
  - first\_argument\_type, 1072
  - result\_type, 1073
  - second\_argument\_type, 1073
- \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 1085
  - \_GuardedIterator, 1085
  - operator \_RAIter, 1086
  - operator<, 1087
  - operator<=, 1087
  - operator\*, 1086
  - operator++, 1086
- \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1118
  - first, 1123
  - first\_type, 1120
  - make\_pair, 1120
- operator!=, 1121
- operator<, 1121
- operator<=, 1121
- operator>, 1122
- operator>=, 1122
- operator==, 1122
- second, 1123
- second\_type, 1120
- swap, 1120, 1122
- \_\_gnu\_parallel::IteratorTriple< \_Iterator1, \_Iterator2, \_Iterator3, \_IteratorCategory >, 1123
- \_\_gnu\_parallel::Job< \_DifferenceTp >, 1124
  - \_M\_first, 1125
  - \_M\_last, 1125
  - \_M\_load, 1125
- \_\_gnu\_parallel::Less< \_T1, \_T2 >, 1127
  - first\_argument\_type, 1128
  - result\_type, 1128
  - second\_argument\_type, 1128
- \_\_gnu\_parallel::Lexicographic< \_T1, \_T2, \_Compare >, 1129
  - first\_argument\_type, 1130
  - result\_type, 1130
  - second\_argument\_type, 1130
- \_\_gnu\_parallel::LexicographicReverse< \_T1, \_T2, \_Compare >, 1131
  - first\_argument\_type, 1131
  - result\_type, 1132
  - second\_argument\_type, 1132
- \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, 1145
  - \_M\_log\_k, 1147
  - \_delete\_min\_insert, 1146
  - \_get\_min\_source, 1146
  - \_insert\_start, 1146
- \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1147
  - \_delete\_min\_insert, 1148
  - \_get\_min\_source, 1148
  - \_init\_winner, 1149
  - \_insert\_start, 1149
- \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, 1150
  - \_LoserTreeBase, 1151
  - \_M\_comp, 1153
  - \_M\_first\_insert, 1153
  - \_M\_log\_k, 1153
  - \_M\_losers, 1153
  - \_get\_min\_source, 1152
  - \_insert\_start, 1152
  - ~\_LoserTreeBase, 1151
- \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >::\_Loser, 1143
  - \_M\_key, 1143



- [\\_M\\_source](#), 1143
- [\\_M\\_sup](#), 1144
- [\\_\\_gnu\\_parallel::\\_LoserTreePointer< \\_\\_stable, \\_Tp, \\_Compare >](#), 1154
- [\\_\\_gnu\\_parallel::\\_LoserTreePointer< false, \\_Tp, \\_Compare >](#), 1155
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >](#), 1156
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerBase< \\_Tp, \\_Compare >::\\_Loser](#), 1144
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#), 1157
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguarded< false, \\_Tp, \\_Compare >](#), 1158
- [\\_\\_gnu\\_parallel::\\_LoserTreePointerUnguardedBase< \\_Tp, \\_Compare >](#), 1159
- [\\_\\_gnu\\_parallel::\\_LoserTreeTraits< \\_Tp >](#), 1160
- [\\_M\\_use\\_pointer](#), 1161
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< \\_\\_stable, \\_Tp, \\_Compare >](#), 1161
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguarded< false, \\_Tp, \\_Compare >](#), 1162
- [\\_\\_gnu\\_parallel::\\_LoserTreeUnguardedBase< \\_Tp, \\_Compare >](#), 1163
- [\\_\\_gnu\\_parallel::\\_Multiplies< \\_Tp1, \\_Tp2, \\_Result >](#), 1171
  - [first\\_argument\\_type](#), 1171
  - [result\\_type](#), 1172
  - [second\\_argument\\_type](#), 1172
- [\\_\\_gnu\\_parallel::\\_Nothing](#), 1185
  - [operator\(\)](#), 1185
- [\\_\\_gnu\\_parallel::\\_PMWSSortingData< \\_RAIter >](#), 1189
  - [\\_M\\_num\\_threads](#), 1189
  - [\\_M\\_offsets](#), 1190
  - [\\_M\\_pieces](#), 1190
  - [\\_M\\_samples](#), 1190
  - [\\_M\\_source](#), 1190
  - [\\_M\\_starts](#), 1191
  - [\\_M\\_temporary](#), 1191
- [\\_\\_gnu\\_parallel::\\_Piece< \\_DifferenceTp >](#), 1186
  - [\\_M\\_begin](#), 1186
  - [\\_M\\_end](#), 1186
- [\\_\\_gnu\\_parallel::\\_Plus< \\_Tp1, \\_Tp2, \\_Result >](#), 1187
  - [first\\_argument\\_type](#), 1188
  - [result\\_type](#), 1188
  - [second\\_argument\\_type](#), 1188
- [\\_\\_gnu\\_parallel::\\_PseudoSequence< \\_Tp, \\_DifferenceTp >](#), 1196
  - [\\_PseudoSequence](#), 1196
  - [begin](#), 1197
  - [end](#), 1197
- [\\_\\_gnu\\_parallel::\\_PseudoSequenceIterator< \\_Tp, \\_DifferenceTp >](#), 1197
- [\\_\\_gnu\\_parallel::\\_QSBThreadLocal< \\_RAIter >](#), 1198
  - [\\_M\\_elements\\_leftover](#), 1199
  - [\\_M\\_global](#), 1199
  - [\\_M\\_initial](#), 1200
  - [\\_M\\_leftover\\_parts](#), 1200
  - [\\_M\\_num\\_threads](#), 1200
  - [\\_Piece](#), 1198
  - [\\_QSBThreadLocal](#), 1199
- [\\_\\_gnu\\_parallel::\\_RandomNumber](#), 1201
  - [\\_RandomNumber](#), 1201, 1202
  - [\\_genrand\\_bits](#), 1202
  - [operator\(\)](#), 1202, 1203
- [\\_\\_gnu\\_parallel::\\_RestrictedBoundedConcurrentQueue< \\_Tp >](#), 1206
  - [\\_RestrictedBoundedConcurrentQueue](#), 1206
  - [~\\_RestrictedBoundedConcurrentQueue](#), 1207
  - [pop\\_back](#), 1207
  - [pop\\_front](#), 1207
  - [push\\_front](#), 1207
- [\\_\\_gnu\\_parallel::\\_SamplingSorter< \\_\\_stable, \\_RAIter, \\_StrictWeakOrdering >](#), 1274
- [\\_\\_gnu\\_parallel::\\_SamplingSorter< false, \\_RAIter, \\_StrictWeakOrdering >](#), 1274
- [\\_\\_gnu\\_parallel::\\_Settings](#), 1277
  - [accumulate\\_minimal\\_n](#), 1279
  - [adjacent\\_difference\\_minimal\\_n](#), 1279
  - [cache\\_line\\_size](#), 1279
  - [count\\_minimal\\_n](#), 1279
  - [fill\\_minimal\\_n](#), 1279
  - [find\\_increasing\\_factor](#), 1280
  - [find\\_initial\\_block\\_size](#), 1280
  - [find\\_maximum\\_block\\_size](#), 1280
  - [find\\_scale\\_factor](#), 1280
  - [find\\_sequential\\_search\\_size](#), 1280
  - [for\\_each\\_minimal\\_n](#), 1281
  - [generate\\_minimal\\_n](#), 1281
  - [get](#), 1278
  - [L1\\_cache\\_size](#), 1281
  - [L2\\_cache\\_size](#), 1281
  - [max\\_element\\_minimal\\_n](#), 1281
  - [merge\\_minimal\\_n](#), 1282
  - [merge\\_oversampling](#), 1282
  - [min\\_element\\_minimal\\_n](#), 1282
  - [multiway\\_merge\\_minimal\\_k](#), 1282
  - [multiway\\_merge\\_minimal\\_n](#), 1282
  - [multiway\\_merge\\_oversampling](#), 1283
  - [nth\\_element\\_minimal\\_n](#), 1283
  - [partial\\_sort\\_minimal\\_n](#), 1283
  - [partial\\_sum\\_dilation](#), 1283
  - [partial\\_sum\\_minimal\\_n](#), 1283
  - [partition\\_chunk\\_share](#), 1284
  - [partition\\_chunk\\_size](#), 1284
  - [partition\\_minimal\\_n](#), 1284
  - [qsb\\_steals](#), 1284
  - [random\\_shuffle\\_minimal\\_n](#), 1284
  - [replace\\_minimal\\_n](#), 1285

[search\\_minimal\\_n](#), 1285  
[set](#), 1278  
[set\\_difference\\_minimal\\_n](#), 1285  
[set\\_intersection\\_minimal\\_n](#), 1285  
[set\\_symmetric\\_difference\\_minimal\\_n](#), 1285  
[set\\_union\\_minimal\\_n](#), 1286  
[sort\\_minimal\\_n](#), 1286  
[sort\\_mwms\\_oversampling](#), 1286  
[sort\\_qs\\_num\\_samples\\_preset](#), 1286  
[sort\\_qsb\\_base\\_case\\_maximal\\_n](#), 1286  
[TLB\\_size](#), 1287  
[transform\\_minimal\\_n](#), 1287  
[unique\\_copy\\_minimal\\_n](#), 1287  
[\\_\\_gnu\\_parallel::\\_\\_SplitConsistently< \\_\\_exact, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 1289  
[\\_\\_gnu\\_parallel::\\_\\_SplitConsistently< false, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 1289  
[\\_\\_gnu\\_parallel::\\_\\_SplitConsistently< true, \\_RAIter, \\_Compare, \\_SortingPlacesIterator >](#), 1289  
[\\_\\_gnu\\_parallel::\\_\\_accumulate\\_binop\\_reduct< \\_BinOp >](#), 862  
[\\_\\_gnu\\_parallel::\\_\\_accumulate\\_selector< \\_It >](#), 863  
[\\_M\\_finish\\_iterator](#), 864  
[operator\(\)](#), 863  
[\\_\\_gnu\\_parallel::\\_\\_adjacent\\_difference\\_selector< \\_It >](#), 865  
[\\_M\\_finish\\_iterator](#), 865  
[\\_\\_gnu\\_parallel::\\_\\_adjacent\\_find\\_selector](#), 866  
[\\_M\\_sequential\\_algorithm](#), 867  
[operator\(\)](#), 867  
[\\_\\_gnu\\_parallel::\\_\\_binder1st< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#), 886  
[argument\\_type](#), 887  
[result\\_type](#), 887  
[\\_\\_gnu\\_parallel::\\_\\_binder2nd< \\_Operation, \\_FirstArgumentType, \\_SecondArgumentType, \\_ResultType >](#), 888  
[argument\\_type](#), 889  
[result\\_type](#), 889  
[\\_\\_gnu\\_parallel::\\_\\_count\\_if\\_selector< \\_It, \\_Diff >](#), 895  
[\\_M\\_finish\\_iterator](#), 896  
[operator\(\)](#), 896  
[\\_\\_gnu\\_parallel::\\_\\_count\\_selector< \\_It, \\_Diff >](#), 897  
[\\_M\\_finish\\_iterator](#), 898  
[operator\(\)](#), 897  
[\\_\\_gnu\\_parallel::\\_\\_fill\\_selector< \\_It >](#), 919  
[\\_M\\_finish\\_iterator](#), 920  
[operator\(\)](#), 920  
[\\_\\_gnu\\_parallel::\\_\\_find\\_first\\_of\\_selector< \\_FIterator >](#), 921  
[\\_M\\_sequential\\_algorithm](#), 922  
[operator\(\)](#), 922  
[\\_\\_gnu\\_parallel::\\_\\_find\\_if\\_selector](#), 923  
[\\_M\\_sequential\\_algorithm](#), 924  
[operator\(\)](#), 924  
[\\_\\_gnu\\_parallel::\\_\\_for\\_each\\_selector< \\_It >](#), 925  
[\\_M\\_finish\\_iterator](#), 926  
[operator\(\)](#), 925  
[\\_\\_gnu\\_parallel::\\_\\_generate\\_selector< \\_It >](#), 928  
[\\_M\\_finish\\_iterator](#), 929  
[operator\(\)](#), 929  
[\\_\\_gnu\\_parallel::\\_\\_generic\\_find\\_selector](#), 930  
[\\_\\_gnu\\_parallel::\\_\\_generic\\_for\\_each\\_selector< \\_It >](#), 931  
[\\_M\\_finish\\_iterator](#), 932  
[\\_\\_gnu\\_parallel::\\_\\_identity\\_selector< \\_It >](#), 932  
[\\_M\\_finish\\_iterator](#), 934  
[operator\(\)](#), 933  
[\\_\\_gnu\\_parallel::\\_\\_inner\\_product\\_selector< \\_It, \\_It2, \\_Tp >](#), 934  
[\\_M\\_finish\\_iterator](#), 936  
[\\_\\_begin1\\_iterator](#), 936  
[\\_\\_begin2\\_iterator](#), 936  
[\\_\\_inner\\_product\\_selector](#), 935  
[operator\(\)](#), 935  
[\\_\\_gnu\\_parallel::\\_\\_max\\_element\\_reduct< \\_Compare, \\_It >](#), 939  
[\\_\\_gnu\\_parallel::\\_\\_min\\_element\\_reduct< \\_Compare, \\_It >](#), 939  
[\\_\\_gnu\\_parallel::\\_\\_mismatch\\_selector](#), 941  
[\\_M\\_sequential\\_algorithm](#), 941  
[operator\(\)](#), 942  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 945  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_3\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 946  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 946  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_4\\_variant\\_sentinel\\_switch< true, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 947  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< \\_\\_sentinels, \\_\\_stable, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 947  
[\\_\\_gnu\\_parallel::\\_\\_multiway\\_merge\\_k\\_variant\\_sentinel\\_switch< false, \\_\\_stable, \\_RAIterIterator, \\_RAIter3, \\_DifferenceTp, \\_Compare >](#), 948  
[\\_\\_gnu\\_parallel::\\_\\_replace\\_if\\_selector< \\_It, \\_Op, \\_Tp >](#), 965  
[\\_M\\_finish\\_iterator](#), 967  
[\\_\\_new\\_val](#), 967  
[\\_\\_replace\\_if\\_selector](#), 966  
[operator\(\)](#), 966  
[\\_\\_gnu\\_parallel::\\_\\_replace\\_selector< \\_It, \\_Tp >](#), 967  
[\\_M\\_finish\\_iterator](#), 969

- [\\_\\_new\\_val](#), 969
- [\\_\\_replace\\_selector](#), 968
- [operator\(\)](#), 968
- [\\_\\_gnu\\_parallel::\\_\\_transform1\\_selector< \\_It >](#), 970
  - [\\_M\\_finish\\_iterator](#), 971
  - [operator\(\)](#), 971
- [\\_\\_gnu\\_parallel::\\_\\_transform2\\_selector< \\_It >](#), 972
  - [\\_M\\_finish\\_iterator](#), 973
  - [operator\(\)](#), 973
- [\\_\\_gnu\\_parallel::\\_\\_unary\\_negate< \\_Predicate, argument\\_type >](#), 974
  - [argument\\_type](#), 974
  - [result\\_type](#), 975
- [\\_\\_gnu\\_parallel::balanced\\_quicksort\\_tag](#), 1380
  - [\\_\\_get\\_num\\_threads](#), 1381
  - [set\\_num\\_threads](#), 1381
- [\\_\\_gnu\\_parallel::balanced\\_tag](#), 1382
  - [\\_\\_get\\_num\\_threads](#), 1382
  - [set\\_num\\_threads](#), 1382
- [\\_\\_gnu\\_parallel::constant\\_size\\_blocks\\_tag](#), 2275
- [\\_\\_gnu\\_parallel::default\\_parallel\\_tag](#), 2400
  - [\\_\\_get\\_num\\_threads](#), 2401
  - [set\\_num\\_threads](#), 2401
- [\\_\\_gnu\\_parallel::equal\\_split\\_tag](#), 2514
- [\\_\\_gnu\\_parallel::exact\\_tag](#), 2520
  - [\\_\\_get\\_num\\_threads](#), 2521
  - [set\\_num\\_threads](#), 2521
- [\\_\\_gnu\\_parallel::find\\_tag](#), 2538
- [\\_\\_gnu\\_parallel::growing\\_blocks\\_tag](#), 2631
- [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_exact\\_tag](#), 3086
  - [\\_\\_get\\_num\\_threads](#), 3086
  - [set\\_num\\_threads](#), 3086
- [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_sampling\\_tag](#), 3087
  - [\\_\\_get\\_num\\_threads](#), 3087
  - [set\\_num\\_threads](#), 3088
- [\\_\\_gnu\\_parallel::multiway\\_mergesort\\_tag](#), 3088
  - [\\_\\_get\\_num\\_threads](#), 3089
  - [set\\_num\\_threads](#), 3089
- [\\_\\_gnu\\_parallel::omp\\_loop\\_static\\_tag](#), 3190
  - [\\_\\_get\\_num\\_threads](#), 3191
  - [set\\_num\\_threads](#), 3191
- [\\_\\_gnu\\_parallel::omp\\_loop\\_tag](#), 3192
  - [\\_\\_get\\_num\\_threads](#), 3192
  - [set\\_num\\_threads](#), 3192
- [\\_\\_gnu\\_parallel::parallel\\_tag](#), 3234
  - [\\_\\_get\\_num\\_threads](#), 3235
  - [parallel\\_tag](#), 3235
  - [set\\_num\\_threads](#), 3235
- [\\_\\_gnu\\_parallel::quicksort\\_tag](#), 3302
  - [\\_\\_get\\_num\\_threads](#), 3302
  - [set\\_num\\_threads](#), 3302
- [\\_\\_gnu\\_parallel::sampling\\_tag](#), 3404
  - [\\_\\_get\\_num\\_threads](#), 3404
  - [set\\_num\\_threads](#), 3404
- [\\_\\_gnu\\_parallel::sequential\\_tag](#), 3416
- [\\_\\_gnu\\_parallel::unbalanced\\_tag](#), 3692
  - [\\_\\_get\\_num\\_threads](#), 3693
  - [set\\_num\\_threads](#), 3693
- [\\_\\_gnu\\_pbds](#), 584
  - [\\_\\_gnu\\_pbds::associative\\_tag](#), 1328
  - [\\_\\_gnu\\_pbds::basic\\_branch< Key, Mapped, Tag, Node\\_Update, Policy\\_Tl, \\_Alloc >](#), 1383
  - [\\_\\_gnu\\_pbds::basic\\_branch\\_tag](#), 1385
  - [\\_\\_gnu\\_pbds::basic\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Resize\\_Policy, Store\\_Hash, Tag, Policy\\_Tl, \\_Alloc >](#), 1480
  - [\\_\\_gnu\\_pbds::basic\\_hash\\_tag](#), 1481
  - [\\_\\_gnu\\_pbds::basic\\_invalidation\\_guarantee](#), 1536
  - [\\_\\_gnu\\_pbds::binary\\_heap\\_tag](#), 2132
  - [\\_\\_gnu\\_pbds::binomial\\_heap\\_tag](#), 2148
  - [\\_\\_gnu\\_pbds::cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger< External\\_Load\\_Access, Size\\_Type >](#), 2175
    - [cc\\_hash\\_max\\_collision\\_check\\_resize\\_trigger](#), 2177
    - [external\\_load\\_access](#), 2177
    - [get\\_load](#), 2177
    - [is\\_grow\\_needed](#), 2177
    - [is\\_resize\\_needed](#), 2177
    - [notify\\_cleared](#), 2178
    - [notify\\_erase\\_search\\_collision](#), 2178
    - [notify\\_erase\\_search\\_end](#), 2178
    - [notify\\_erase\\_search\\_start](#), 2178
    - [notify\\_erased](#), 2179
    - [notify\\_externally\\_resized](#), 2179
    - [notify\\_find\\_search\\_collision](#), 2179
    - [notify\\_find\\_search\\_end](#), 2179
    - [notify\\_find\\_search\\_start](#), 2180
    - [notify\\_insert\\_search\\_collision](#), 2180
    - [notify\\_insert\\_search\\_end](#), 2180
    - [notify\\_insert\\_search\\_start](#), 2180
    - [notify\\_inserted](#), 2181
    - [notify\\_resized](#), 2181
    - [set\\_load](#), 2181
  - [\\_\\_gnu\\_pbds::cc\\_hash\\_table< Key, Mapped, Hash\\_Fn, Eq\\_Fn, Comb\\_Hash\\_Fn, Resize\\_Policy, Store\\_Hash, \\_Alloc >](#), 2182
    - [cc\\_hash\\_table](#), 2183–2186
  - [\\_\\_gnu\\_pbds::cc\\_hash\\_tag](#), 2187
  - [\\_\\_gnu\\_pbds::container\\_error](#), 2292
    - [what](#), 2292
  - [\\_\\_gnu\\_pbds::container\\_tag](#), 2293
  - [\\_\\_gnu\\_pbds::container\\_traits< Cntnr >](#), 2293
    - [erase\\_can\\_throw](#), 2294
    - [order\\_preserving](#), 2294
    - [reverse\\_iteration](#), 2294
    - [split\\_join\\_can\\_throw](#), 2294
  - [\\_\\_gnu\\_pbds::container\\_traits\\_base< \\_Tag >](#), 2294
  - [\\_\\_gnu\\_pbds::container\\_traits\\_base< binary\\_heap\\_tag >](#), 2295

[\\_\\_gnu\\_pbds::container\\_traits\\_base< binomial\\_heap\\_tag >](#), [2295](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< cc\\_hash\\_tag >](#), [2296](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< gp\\_hash\\_tag >](#), [2296](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< list\\_update\\_tag >](#), [2297](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< ov\\_tree\\_tag >](#), [2297](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< pairing\\_heap\\_tag >](#), [2298](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< pat\\_trie\\_tag >](#), [2298](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rb\\_tree\\_tag >](#), [2299](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< rc\\_binomial\\_heap\\_tag >](#), [2299](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< splay\\_tree\\_tag >](#), [2300](#)  
[\\_\\_gnu\\_pbds::container\\_traits\\_base< thin\\_heap\\_tag >](#), [2300](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#), [2103](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_const\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#), [2104](#)  
[const\\_reference](#), [2105](#)  
[difference\\_type](#), [2105](#)  
[get\\_l\\_child](#), [2107](#)  
[get\\_metadata](#), [2107](#)  
[get\\_r\\_child](#), [2107](#)  
[iterator\\_category](#), [2106](#)  
[metadata\\_const\\_reference](#), [2106](#)  
[metadata\\_type](#), [2106](#)  
[operator!=](#), [2108](#)  
[operator\\*](#), [2108](#)  
[operator==](#), [2108](#)  
[reference](#), [2106](#)  
[value\\_type](#), [2107](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_it\\_< Node\\_Pointer, Value\\_Type, Pointer, Const\\_Pointer, Reference, Const\\_Reference, Is\\_Forward\\_Iterator, \\_Alloc >](#), [2109](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_node\\_it\\_< Node, Const\\_Iterator, Iterator, \\_Alloc >](#), [2111](#)  
[const\\_reference](#), [2112](#)  
[difference\\_type](#), [2112](#)  
[get\\_l\\_child](#), [2113](#)  
[get\\_metadata](#), [2113](#)  
[get\\_r\\_child](#), [2114](#)  
[iterator\\_category](#), [2112](#)  
[metadata\\_const\\_reference](#), [2112](#)  
[metadata\\_type](#), [2112](#)  
[operator!=](#), [2114](#)  
[operator\\*](#), [2114](#)  
[operator==](#), [2114](#)  
[reference](#), [2113](#)  
[value\\_type](#), [2113](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#), [2115](#)  
[node\\_const\\_iterator](#), [2116](#)  
[\\_\\_gnu\\_pbds::detail::bin\\_search\\_tree\\_traits< Key, null\\_type, Cmp\\_Fn, Node\\_Update, Node, \\_Alloc >](#), [2116](#)  
[node\\_const\\_iterator](#), [2117](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [2120](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#), [2122](#)  
[binary\\_heap\\_const\\_iterator\\_](#), [2125](#)  
[const\\_pointer](#), [2123](#)  
[const\\_reference](#), [2124](#)  
[difference\\_type](#), [2124](#)  
[iterator\\_category](#), [2124](#)  
[operator!=](#), [2126](#)  
[operator\\*](#), [2126](#)  
[operator->](#), [2126](#)  
[operator==](#), [2127](#)  
[pointer](#), [2124](#)  
[reference](#), [2125](#)  
[value\\_type](#), [2125](#)  
[\\_\\_gnu\\_pbds::detail::binary\\_heap\\_point\\_const\\_iterator\\_< Value\\_Type, Entry, Simple, \\_Alloc >](#), [2128](#)  
[binary\\_heap\\_point\\_const\\_iterator\\_](#), [2130](#), [2131](#)  
[const\\_pointer](#), [2129](#)  
[const\\_reference](#), [2129](#)  
[difference\\_type](#), [2129](#)  
[iterator\\_category](#), [2129](#)  
[operator!=](#), [2131](#)  
[operator\\*](#), [2131](#)  
[operator->](#), [2131](#)  
[operator==](#), [2132](#)  
[pointer](#), [2130](#)  
[reference](#), [2130](#)  
[value\\_type](#), [2130](#)  
[\\_\\_gnu\\_pbds::detail::binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [2143](#)  
[\\_\\_gnu\\_pbds::detail::binomial\\_heap\\_base< Value\\_Type, Cmp\\_Fn, \\_Alloc >](#), [2145](#)  
[\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Cltr, \\_Alloc >](#), [2171](#)  
[\\_\\_gnu\\_pbds::detail::branch\\_policy< Node\\_Cltr, Node\\_Itr, \\_Alloc >](#), [2170](#)  
[\\_\\_gnu\\_pbds::detail::cc\\_ht\\_map< Key, Mapped, Hash\\_Fn, Eq\\_Fn, \\_Alloc, Store\\_Hash, Comb\\_Hash\\_Fn, Resize\\_Policy >](#), [2188](#)  
[empty](#), [2191](#)  
[get\\_comb\\_hash\\_fn](#), [2191](#)

- get\_eq\_fn, [2191](#)
- get\_hash\_fn, [2192](#)
- get\_resize\_policy, [2192](#)
- \_\_gnu\_pbds::detail::cond\_dealtor< Entry, \_Alloc >, [2263](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, Tag, Policy\_TI >, [2277](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [2281](#)
- type, [2282](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [2282](#)
- type, [2283](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [2283](#)
- type, [2283](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >, [2284](#)
- type, [2284](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, pat\_trie\_tag, Policy\_TI >, [2284](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >, [2285](#)
- type, [2285](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >, [2286](#)
- type, [2286](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >, [2286](#)
- type, [2287](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >, [2287](#)
- type, [2288](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >, [2288](#)
- type, [2288](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI >, [2289](#)
- type, [2289](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI >, [2289](#)
- type, [2290](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, rb\_tree\_tag, Policy\_TI >, [2290](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI >, [2291](#)
- type, [2291](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [2278](#)
- type, [2278](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [2278](#)
- type, [2279](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [2279](#)
- type, [2280](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [2280](#)
- type, [2280](#)
- \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [2281](#)
- type, [2281](#)
- \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [2395](#)
- type, [2396](#)
- \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [2399](#)
- type, [2399](#)
- \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [2400](#)
- type, [2400](#)
- \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >, [2402](#)
- type, [2402](#)
- \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_Hash\_Fn >, [2402](#)
- type, [2403](#)
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< Key >, [2403](#)
- \_\_gnu\_pbds::detail::default\_trie\_access\_traits< std::basic\_string< Char, Char\_Traits, std::allocator< char > > >, [2404](#)
- type, [2404](#)
- \_\_gnu\_pbds::detail::default\_update\_policy, [2404](#)
- type, [2405](#)
- \_\_gnu\_pbds::detail::dumnode\_const\_iterator< Key, Data, \_Alloc >, [2459](#)
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, No\_Throw >, [2511](#)
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >, [2511](#)
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, false >::type, [3684](#)
- \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, [2512](#)
- type, [2512](#)

- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, No\_Throw >, [2513](#)
- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, false >, [2513](#)
- \_\_gnu\_pbds::detail::entry\_pred< \_VTp, Pred, \_Alloc, true >, [2513](#)
- \_\_gnu\_pbds::detail::eq\_by\_less< Key, Cmp\_Fn >, [2514](#)
- \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2621](#)
- empty, [2623](#)
- get\_comb\_probe\_fn, [2624](#)
- get\_eq\_fn, [2624](#)
- get\_hash\_fn, [2625](#)
- get\_probe\_fn, [2625](#)
- get\_resize\_policy, [2626](#)
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, Store\_Hash >, [2662](#)
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, false >, [2663](#)
- \_\_gnu\_pbds::detail::hash\_eq\_fn< Key, Eq\_Fn, \_Alloc, true >, [2663](#)
- \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, Hold\_Size >, [2668](#)
- \_\_gnu\_pbds::detail::hash\_load\_check\_resize\_trigger\_size\_base< Size\_Type, true >, [2669](#)
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap< Value\_Type, Cmp\_Fn, Node\_Metadata, \_Alloc >, [2798](#)
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2800](#)
- const\_pointer, [2801](#)
- const\_reference, [2801](#)
- difference\_type, [2801](#)
- iterator\_category, [2801](#)
- left\_child\_next\_sibling\_heap\_const\_iterator\_, [2802](#), [2803](#)
- operator!=, [2803](#)
- operator\*, [2803](#)
- operator->, [2804](#)
- operator==, [2804](#)
- pointer, [2802](#)
- reference, [2802](#)
- value\_type, [2802](#)
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node< \_Value, \_Metadata, \_Alloc >, [2805](#)
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [2806](#)
- const\_pointer, [2807](#)
- const\_reference, [2807](#)
- difference\_type, [2807](#)
- iterator\_category, [2807](#)
- left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_, [2808](#), [2809](#)
- operator!=, [2809](#)
- operator\*, [2809](#)
- operator->, [2809](#)
- operator==, [2810](#)
- pointer, [2808](#)
- reference, [2808](#)
- value\_type, [2808](#)
- \_\_gnu\_pbds::detail::lu\_counter\_metadata< Size\_Type >, [2886](#)
- \_\_gnu\_pbds::detail::lu\_counter\_policy\_base< Size\_Type >, [2889](#)
- \_\_gnu\_pbds::detail::lu\_map< Key, Mapped, Eq\_Fn, \_Alloc, Update\_Policy >, [2890](#)
- \_\_gnu\_pbds::detail::mask\_based\_range\_hashing< Size\_Type >, [2936](#)
- \_\_gnu\_pbds::detail::maybe\_null\_type< Key, Mapped, \_Alloc, Store\_Hash >, [2950](#)
- \_\_gnu\_pbds::detail::maybe\_null\_type< Key, null\_type, \_Alloc, Store\_Hash >, [2950](#)
- \_\_gnu\_pbds::detail::mod\_based\_range\_hashing< Size\_Type >, [2973](#)
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, Mapped >, [3100](#)
- \_\_gnu\_pbds::detail::no\_throw\_copies< Key, null\_type >, [3100](#)
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3208](#)
- node\_begin, [3210](#), [3211](#)
- node\_end, [3211](#)
- \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >::cond\_dtor< Size\_Type >, [2264](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_const\_it< Value\_Type, Metadata\_Type, \_Alloc >, [3212](#)
- get\_l\_child, [3213](#)
- get\_r\_child, [3213](#)
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it< Value\_Type, Metadata\_Type, \_Alloc >, [3214](#)
- get\_l\_child, [3215](#)
- get\_r\_child, [3215](#)
- operator\*, [3215](#)
- \_\_gnu\_pbds::detail::pairing\_heap< Value\_Type, Cmp\_Fn, \_Alloc >, [3231](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base, [3250](#)
- node\_type, [3251](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::CIter< Node, Leaf, Metadata, Is\_Forward\_Iterator >, [1053](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::Head< \_ATraits, Metadata >, [1108](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >, [1109](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >::const\_iterator, [2267](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Inode< \_ATraits, Metadata >::iterator, [2792](#)



[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Iter< Node, Leaf, Head, Inode, Is\\_Forward\\_Iterator >, 1116](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Leaf< \\_ATraits, Metadata >, 1126](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< Metadata, \\_Alloc >, 1167](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Metadata< null\\_type, \\_Alloc >, 1167](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_base< \\_ATraits, Metadata >, 1172](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_citer< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >, 1173](#)  
[get\\_child, 1175](#)  
[get\\_metadata, 1175](#)  
[metadata\\_const\\_reference, 1175](#)  
[metadata\\_type, 1175](#)  
[num\\_children, 1176](#)  
[operator!=, 1176](#)  
[operator\\*, 1176](#)  
[operator==, 1176](#)  
[valid\\_prefix, 1177](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_base::\\_Node\\_iter< Node, Leaf, Head, Inode, \\_Clterator, Iterator, \\_Alloc >, 1178](#)  
[get\\_child, 1180](#)  
[get\\_metadata, 1180](#)  
[metadata\\_const\\_reference, 1180](#)  
[metadata\\_type, 1180](#)  
[num\\_children, 1181](#)  
[operator!=, 1181](#)  
[operator\\*, 1181](#)  
[operator==, 1181](#)  
[valid\\_prefix, 1182](#)  
[\\_\\_gnu\\_pbds::detail::pat\\_trie\\_map< Key, Mapped, Node\\_And\\_It\\_Traits, \\_Alloc >, 3251](#)  
[node\\_begin, 3254](#)  
[node\\_end, 3254](#)  
[node\\_type, 3254](#)  
[\\_\\_gnu\\_pbds::detail::probe\\_fn\\_base< \\_Alloc >, 3289](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, Store\\_Hash >, 3308](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, false >, 3308](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Hash\\_Fn, true >, 3309](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, false >, 3310](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_hash\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Hash\\_Fn, true >, 3311](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, Store\\_Hash >, 3311](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, false >, 3312](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, Hash\\_Fn, \\_Alloc, Comb\\_Probe\\_Fn, Probe\\_Fn, true >, 3313](#)  
[\\_\\_gnu\\_pbds::detail::ranged\\_probe\\_fn< Key, null\\_type, \\_Alloc, Comb\\_Probe\\_Fn, null\\_type, false >, 3313](#)  
[\\_\\_gnu\\_pbds::detail::rb\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >, 3327](#)  
[node\\_begin, 3331](#)  
[node\\_end, 3331](#)  
[\\_\\_gnu\\_pbds::detail::rb\\_tree\\_node< Value\\_Type, Metadata, \\_Alloc >, 3332](#)  
[\\_\\_gnu\\_pbds::detail::rc< \\_Node, \\_Alloc >, 3334](#)  
[\\_\\_gnu\\_pbds::detail::rc\\_binomial\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >, 3334](#)  
[\\_\\_gnu\\_pbds::detail::rebind\\_traits< \\_Alloc, T >, 3337](#)  
[\\_\\_gnu\\_pbds::detail::resize\\_policy< \\_Tp >, 3368](#)  
[\\_\\_gnu\\_pbds::detail::select\\_value\\_type< Key, Mapped >, 3410](#)  
[\\_\\_gnu\\_pbds::detail::select\\_value\\_type< Key, null\\_type >, 3411](#)  
[\\_\\_gnu\\_pbds::detail::splay\\_tree\\_map< Key, Mapped, Cmp\\_Fn, Node\\_And\\_It\\_Traits, \\_Alloc >, 3487](#)  
[node\\_begin, 3490](#)  
[node\\_end, 3490, 3491](#)  
[\\_\\_gnu\\_pbds::detail::splay\\_tree\\_node< Value\\_Type, Metadata, \\_Alloc >, 3491](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th, Store\\_Hash >, 3547](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_data< \\_Tv, \\_Th, false >, 3548](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_hash< \\_Th >, 3549](#)  
[\\_\\_gnu\\_pbds::detail::stored\\_value< \\_Tv >, 3550](#)  
[\\_\\_gnu\\_pbds::detail::synth\\_access\\_traits< Type\\_Traits, Set, \\_ATraits >, 3574](#)  
[\\_\\_gnu\\_pbds::detail::thin\\_heap< Value\\_Type, Cmp\\_Fn, \\_Alloc >, 3580](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, \\_BTp >, 3637](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, false >, 3637](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_metadata\\_helper< Node\\_Update, true >, 3638](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_node\\_metadata\\_dispatch< Key, Data, Cmp\\_Fn, Node\\_Update, \\_Alloc >, 3638](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Data, Cmp\\_Fn, Node\\_Update, Tag, \\_Alloc >, 3642](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, ov\\_tree\\_tag, \\_Alloc >, 3643](#)  
[node\\_const\\_iterator, 3643](#)  
[\\_\\_gnu\\_pbds::detail::tree\\_traits< Key, Mapped, Cmp\\_Fn, Node\\_Update, rb\\_tree\\_tag, \\_Alloc >, 3644](#)  
[node\\_const\\_iterator, 3645](#)

- \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3645](#)
  - node\_const\_iterator, [3646](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [3647](#)
  - node\_const\_iterator, [3647](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [3648](#)
  - node\_const\_iterator, [3649](#)
- \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3649](#)
  - node\_const\_iterator, [3650](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, BTp >, [3653](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, false >, [3654](#)
- \_\_gnu\_pbds::detail::trie\_metadata\_helper< Node\_Update, true >, [3654](#)
- \_\_gnu\_pbds::detail::trie\_node\_metadata\_dispatch< Key, Data, Cmp\_Fn, Node\_Update, \_Alloc >, [3654](#)
- \_\_gnu\_pbds::detail::trie\_policy\_base< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, [3658](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Data, ATraits, Node\_Update, Tag, \_Alloc >, [3668](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3668](#)
  - node\_const\_iterator, [3669](#)
  - node\_update, [3669](#)
  - synth\_access\_traits, [3669](#)
- \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3670](#)
  - node\_const\_iterator, [3670](#)
  - node\_update, [3670](#)
  - synth\_access\_traits, [3671](#)
- \_\_gnu\_pbds::detail::types\_traits< Key, Mapped, \_Alloc, Store\_Hash >, [3687](#)
- \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type >, [2442](#)
  - operator(), [2443](#)
- \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type >, [2443](#)
  - operator(), [2444](#)
- \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, [2613](#)
  - gp\_hash\_table, [2615–2619](#)
- \_\_gnu\_pbds::gp\_hash\_tag, [2620](#)
- \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, [2664](#)
  - hash\_exponential\_size\_policy, [2664](#)
- \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2665](#)
  - external\_load\_access, [2666](#)
  - get\_loads, [2667](#)
  - hash\_load\_check\_resize\_trigger, [2667](#)
  - notify\_cleared, [2667](#)
  - notify\_inserted, [2667](#)
  - notify\_resized, [2668](#)
  - set\_loads, [2668](#)
- \_\_gnu\_pbds::hash\_prime\_size\_policy, [2675](#)
  - hash\_prime\_size\_policy, [2675](#)
  - size\_type, [2675](#)
- \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [2678](#)
  - get\_actual\_size, [2680](#)
  - get\_new\_size, [2680](#)
  - get\_size\_policy, [2680](#)
  - get\_trigger\_policy, [2681](#)
  - hash\_standard\_resize\_policy, [2679](#)
  - resize, [2681](#)
- \_\_gnu\_pbds::insert\_error, [2696](#)
  - what, [2696](#)
- \_\_gnu\_pbds::join\_error, [2797](#)
  - what, [2798](#)
- \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type >, [2824](#)
  - operator(), [2825](#)
- \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, [2859](#)
  - list\_update, [2860](#)
- \_\_gnu\_pbds::list\_update\_tag, [2861](#)
- \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, [2887](#)
  - max\_count, [2889](#)
  - metadata\_reference, [2888](#)
  - metadata\_type, [2888](#)
  - operator(), [2889](#)
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, [2892](#)
  - metadata\_reference, [2893](#)
  - metadata\_type, [2893](#)
  - operator(), [2893](#)
- \_\_gnu\_pbds::null\_node\_update< \_Tp1, \_Tp2, \_Tp3, \_Tp4 >, [3109](#)
- \_\_gnu\_pbds::null\_type, [3110](#)
- \_\_gnu\_pbds::ov\_tree\_tag, [3216](#)
- \_\_gnu\_pbds::pairing\_heap\_tag, [3233](#)
- \_\_gnu\_pbds::pat\_trie\_tag, [3255](#)
- \_\_gnu\_pbds::point\_invalidation\_guarantee, [3270](#)
- \_\_gnu\_pbds::priority\_queue< \_Tv, Cmp\_Fn, Tag, \_Alloc >, [3282](#)
- \_\_gnu\_pbds::priority\_queue\_tag, [3288](#)
- \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, [3295](#)
  - operator(), [3296](#)
- \_\_gnu\_pbds::range\_invalidation\_guarantee, [3307](#)
- \_\_gnu\_pbds::rb\_tree\_tag, [3333](#)
- \_\_gnu\_pbds::rc\_binomial\_heap\_tag, [3337](#)
- \_\_gnu\_pbds::resize\_error, [3367](#)
  - what, [3367](#)



- \_\_gnu\_pbds::sample\_probe\_fn, [3383](#)
  - operator(), [3383](#)
  - sample\_probe\_fn, [3383](#)
  - swap, [3384](#)
- \_\_gnu\_pbds::sample\_range\_hashing, [3384](#)
  - notify\_resized, [3385](#)
  - operator(), [3385](#)
  - sample\_range\_hashing, [3385](#)
  - size\_type, [3385](#)
  - swap, [3386](#)
- \_\_gnu\_pbds::sample\_ranged\_hash\_fn, [3386](#)
  - notify\_resized, [3387](#)
  - operator(), [3387](#)
  - sample\_ranged\_hash\_fn, [3386](#), [3387](#)
  - swap, [3387](#)
- \_\_gnu\_pbds::sample\_ranged\_probe\_fn, [3388](#)
- \_\_gnu\_pbds::sample\_resize\_policy, [3388](#)
  - get\_new\_size, [3390](#)
  - is\_resize\_needed, [3390](#)
  - notify\_cleared, [3390](#)
  - notify\_erase\_search\_collision, [3390](#)
  - notify\_erase\_search\_end, [3390](#)
  - notify\_erase\_search\_start, [3390](#)
  - notify\_erased, [3391](#)
  - notify\_find\_search\_collision, [3391](#)
  - notify\_find\_search\_end, [3391](#)
  - notify\_find\_search\_start, [3391](#)
  - notify\_insert\_search\_collision, [3391](#)
  - notify\_insert\_search\_end, [3391](#)
  - notify\_insert\_search\_start, [3392](#)
  - notify\_inserted, [3392](#)
  - notify\_resized, [3392](#)
  - sample\_range\_hashing, [3392](#)
  - sample\_resize\_policy, [3389](#)
  - size\_type, [3389](#)
  - swap, [3392](#)
- \_\_gnu\_pbds::sample\_resize\_trigger, [3393](#)
  - is\_grow\_needed, [3394](#)
  - is\_resize\_needed, [3394](#)
  - notify\_cleared, [3394](#)
  - notify\_erase\_search\_collision, [3394](#)
  - notify\_erase\_search\_end, [3394](#)
  - notify\_erase\_search\_start, [3395](#)
  - notify\_erased, [3395](#)
  - notify\_externally\_resized, [3395](#)
  - notify\_find\_search\_collision, [3395](#)
  - notify\_find\_search\_end, [3395](#)
  - notify\_find\_search\_start, [3395](#)
  - notify\_insert\_search\_collision, [3396](#)
  - notify\_insert\_search\_end, [3396](#)
  - notify\_insert\_search\_start, [3396](#)
  - notify\_inserted, [3396](#)
  - notify\_resized, [3396](#)
  - sample\_range\_hashing, [3396](#)
  - sample\_resize\_trigger, [3394](#)
  - size\_type, [3393](#)
  - swap, [3397](#)
- \_\_gnu\_pbds::sample\_size\_policy, [3397](#)
  - get\_nearest\_larger\_size, [3398](#)
  - get\_nearest\_smaller\_size, [3398](#)
  - sample\_range\_hashing, [3398](#)
  - sample\_size\_policy, [3398](#)
  - size\_type, [3397](#)
  - swap, [3398](#)
- \_\_gnu\_pbds::sample\_tree\_node\_update< Const\_Node\_Itr, Node\_Itr, Cmp\_Fn, \_Alloc >, [3399](#)
- \_\_gnu\_pbds::sample\_trie\_access\_traits, [3399](#)
  - begin, [3400](#)
  - e\_pos, [3400](#)
  - e\_type, [3400](#)
  - end, [3400](#)
- \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, [3401](#)
  - operator(), [3401](#)
  - sample\_trie\_node\_update, [3401](#)
- \_\_gnu\_pbds::sample\_update\_policy, [3402](#)
  - metadata\_type, [3402](#)
  - operator(), [3403](#)
  - sample\_update\_policy, [3402](#)
  - swap, [3403](#)
- \_\_gnu\_pbds::sequence\_tag, [3415](#)
- \_\_gnu\_pbds::splay\_tree\_tag, [3492](#)
- \_\_gnu\_pbds::string\_tag, [3551](#)
- \_\_gnu\_pbds::thin\_heap\_tag, [3582](#)
- \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, [3634](#)
  - cmp\_fn, [3635](#)
  - tree, [3636](#)
- \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, Cmp\_Fn, \_Alloc >, [3639](#)
  - find\_by\_order, [3640](#), [3641](#)
  - operator(), [3641](#)
  - order\_of\_key, [3641](#)
- \_\_gnu\_pbds::tree\_tag, [3642](#)
- \_\_gnu\_pbds::trie< Key, Mapped, ATraits, Tag, Node\_Update, \_Alloc >, [3651](#)
  - access\_traits, [3652](#)
  - trie, [3652](#), [3653](#)
- \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, [3655](#)
  - find\_by\_order, [3657](#)
  - operator(), [3657](#)
  - order\_of\_key, [3657](#)
  - order\_of\_prefix, [3658](#)
- \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, ATraits, \_Alloc >, [3660](#)
  - a\_const\_iterator, [3662](#)
  - access\_traits, [3662](#)

- allocator\_type, 3662
- operator(), 3663
- prefix\_range, 3663, 3664
- size\_type, 3662
- \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3664
- begin, 3666
- const\_iterator, 3665
- e\_pos, 3666
- e\_type, 3665
- end, 3666
- \_\_gnu\_pbds::trie\_tag, 3667
- \_\_gnu\_pbds::trivial\_iterator\_tag, 3671
- \_\_gnu\_sequential, 586
- \_\_heap\_select
  - std, 694
- \_\_init\_winner
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1149
- \_\_inner\_product\_selector
  - \_\_gnu\_parallel::\_\_inner\_product\_selector< \_It, \_It2, \_Tp >, 935
- \_\_inplace\_stable\_sort
  - std, 694
- \_\_insert\_start
  - \_\_gnu\_parallel::LoserTree< \_\_stable, \_Tp, \_Compare >, 1146
  - \_\_gnu\_parallel::LoserTree< false, \_Tp, \_Compare >, 1149
  - \_\_gnu\_parallel::LoserTreeBase< \_Tp, \_Compare >, 1152
- \_\_insertion\_sort
  - std, 694
- \_\_int\_traits
  - \_\_gnu\_cxx, 499
- \_\_introsort\_loop
  - std, 695
- \_\_invoke
  - Utilities, 474
- \_\_ioinit
  - std, 793
- \_\_is\_sorted
  - \_\_gnu\_parallel, 546
- \_\_iterator\_category
  - Iterators, 137
- \_\_lg
  - std, 695
- \_\_match\_flag
  - std::regex\_constants, 840
- \_\_median
  - SGL, 393, 394
- \_\_median\_of\_three\_iterators
  - \_\_gnu\_parallel, 547
- \_\_merge\_adaptive
  - std, 695
- \_\_merge\_advance
  - \_\_gnu\_parallel, 547
- \_\_merge\_advance\_movc
  - \_\_gnu\_parallel, 548
- \_\_merge\_advance\_usual
  - \_\_gnu\_parallel, 549
- \_\_merge\_without\_buffer
  - std, 696
- \_\_move\_median\_to\_first
  - std, 696
- \_\_move\_merge
  - std, 696
- \_\_move\_merge\_adaptive
  - std, 697
- \_\_move\_merge\_adaptive\_backward
  - std, 697
- \_\_new\_val
  - \_\_gnu\_parallel::\_\_replace\_if\_selector< \_It, \_Op, \_Tp >, 967
  - \_\_gnu\_parallel::\_\_replace\_selector< \_It, \_Tp >, 969
- \_\_num\_bitmaps
  - \_\_gnu\_cxx::\_\_detail, 513
- \_\_num\_blocks
  - \_\_gnu\_cxx::\_\_detail, 514
- \_\_num\_get\_type
  - std::basic\_ios< \_CharT, \_Traits >, 1540
  - std::basic\_ofstream< \_CharT, \_Traits >, 1745
  - std::basic\_ostream< \_CharT, \_Traits >, 1790
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1834
- \_\_num\_put\_type
  - std::basic\_fstream< \_CharT, \_Traits >, 1420
  - std::basic\_ifstream< \_CharT, \_Traits >, 1488
  - std::basic\_ios< \_CharT, \_Traits >, 1541
  - std::basic\_iostream< \_CharT, \_Traits >, 1576
  - std::basic\_istream< \_CharT, \_Traits >, 1640
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1692
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2039
- \_\_parallel\_merge\_advance
  - \_\_gnu\_parallel, 549, 550
- \_\_parallel\_nth\_element
  - \_\_gnu\_parallel, 551
- \_\_parallel\_partial\_sort
  - \_\_gnu\_parallel, 551
- \_\_parallel\_partial\_sum
  - \_\_gnu\_parallel, 552
- \_\_parallel\_partial\_sum\_basecase
  - \_\_gnu\_parallel, 552
- \_\_parallel\_partial\_sum\_linear
  - \_\_gnu\_parallel, 553
- \_\_parallel\_partition

- \_\_gnu\_parallel, 554
- \_\_parallel\_random\_shuffle
  - \_\_gnu\_parallel, 554
- \_\_parallel\_random\_shuffle\_drs
  - \_\_gnu\_parallel, 555
- \_\_parallel\_random\_shuffle\_drs\_pu
  - \_\_gnu\_parallel, 555
- \_\_parallel\_sort
  - \_\_gnu\_parallel, 556–560
- \_\_parallel\_sort\_qs
  - \_\_gnu\_parallel, 561
- \_\_parallel\_sort\_qs\_conquer
  - \_\_gnu\_parallel, 562
- \_\_parallel\_sort\_qs\_divide
  - \_\_gnu\_parallel, 562
- \_\_parallel\_sort\_qsb
  - \_\_gnu\_parallel, 563
- \_\_parallel\_unique\_copy
  - \_\_gnu\_parallel, 563, 564
- \_\_partition
  - std, 697, 698
- \_\_polynomial
  - std::regex\_constants, 848
- \_\_ptr\_rebind
  - std, 687
- \_\_qsb\_conquer
  - \_\_gnu\_parallel, 564
- \_\_qsb\_divide
  - \_\_gnu\_parallel, 565
- \_\_qsb\_local\_sort\_with\_helping
  - \_\_gnu\_parallel, 566
- \_\_random\_number\_pow2
  - \_\_gnu\_parallel, 566
- \_\_rd\_log2
  - \_\_gnu\_parallel, 567
- \_\_replace\_if\_selector
  - \_\_gnu\_parallel::\_\_replace\_if\_selector<\_It, \_Op, \_Tp>, 966
- \_\_replace\_selector
  - \_\_gnu\_parallel::\_\_replace\_selector<\_It, \_Tp>, 968
- \_\_reverse
  - std, 698
- \_\_rotate
  - stl\_algo.h, 4291, 4292
- \_\_rotate\_adaptive
  - std, 699
- \_\_round\_up\_to\_pow2
  - \_\_gnu\_parallel, 567
- \_\_sample
  - std, 699
- \_\_search\_n\_aux
  - std, 700
- \_\_search\_template
  - \_\_gnu\_parallel, 567
- \_\_sequential\_multiway\_merge
  - \_\_gnu\_parallel, 568
- \_\_sequential\_random\_shuffle
  - \_\_gnu\_parallel, 569
- \_\_shrink
  - \_\_gnu\_parallel, 569
- \_\_shrink\_and\_double
  - \_\_gnu\_parallel, 570
- \_\_stable\_partition\_adaptive
  - std, 700
- \_\_static\_pointer\_cast
  - \_\_gnu\_cxx, 499
- \_\_streambuf\_type
  - std::basic\_streambuf<\_CharT, \_Traits>, 1890
  - std::wbuffer\_convert<\_Codecvt, \_Elem, \_Tr>, 3908
- \_\_syntax\_option
  - std::regex\_constants, 840
- \_\_umap\_traits
  - std, 687
- \_\_ummap\_traits
  - std, 687
- \_\_umset\_traits
  - std, 687
- \_\_unguarded\_insertion\_sort
  - std, 701
- \_\_unguarded\_linear\_insert
  - std, 701
- \_\_unguarded\_partition
  - std, 701
- \_\_unguarded\_partition\_pivot
  - std, 702
- \_\_unique\_copy
  - std, 702, 703
- \_\_uset\_traits
  - std, 687
- \_\_valid\_range
  - \_\_gnu\_debug, 523, 524
- \_\_valid\_range\_aux
  - \_\_gnu\_debug, 524
- \_\_verbose\_terminate\_handler
  - Exceptions, 89
- \_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string<\_CharT, \_Traits, \_Alloc, \_Base>, 979–982
- \_\_yield
  - \_\_gnu\_parallel, 570
- ~\_LoserTreeBase
  - \_\_gnu\_parallel::\_LoserTreeBase<\_Tp, \_Compare>, 1151
- ~\_RestrictedBoundedConcurrentQueue
  - \_\_gnu\_parallel::\_RestrictedBoundedConcurrentQueue<\_Tp>, 1207
- ~\_Safe\_sequence\_base
  - \_\_gnu\_debug::\_Safe\_sequence\_base, 1262

- ~\_Safe\_unordered\_container\_base
  - \_\_gnu\_debug::\_Safe\_unordered\_container\_base, 1270
- ~\_\_allocated\_ptr
  - std::\_\_allocated\_ptr< \_Alloc >, 879
- ~\_\_versa\_string
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 983
- ~any
  - std::experimental::fundamentals\_v1::any, 1324
- ~auto\_ptr
  - std::auto\_ptr< \_Tp >, 1364
- ~basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1389
- ~basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 1424
- ~basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 1492
- ~basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 1545
- ~basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 1579
- ~basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 1643
- ~basic\_istreamstream
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1696
- ~basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 1749
- ~basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 1794
- ~basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1838
- ~basic\_regex
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1879
- ~basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, 1892
- ~basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1919
- ~basic\_stringstream
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2043
- ~collate
  - std::collate< \_CharT >, 2244
- ~ctype
  - std::ctype< char >, 2322
  - std::ctype< wchar\_t >, 2339
- ~deque
  - std::deque< \_Tp, \_Alloc >, 2414
- ~facet
  - std::locale::facet, 2536
- ~forward\_list
  - std::forward\_list< \_Tp, \_Alloc >, 2557
- ~gslice
  - Numeric Arrays, 269
- ~ios\_base
  - std::ios\_base, 2710
- ~list
  - std::list< \_Tp, \_Alloc >, 2838
- ~locale
  - std::locale, 2866
- ~map
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2909
- ~match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2942
- ~messages
  - std::messages< \_CharT >, 2966
- ~money\_get
  - std::money\_get< \_CharT, \_InIter >, 2980
- ~money\_put
  - std::money\_put< \_CharT, \_OutIter >, 2986
- ~moneypunct
  - std::moneypunct< \_CharT, \_Intl >, 2993
- ~multimap
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3028
- ~multiset
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3061
- ~num\_get
  - std::num\_get< \_CharT, \_InIter >, 3114
- ~num\_put
  - std::num\_put< \_CharT, \_OutIter >, 3135
- ~numpunct
  - std::numpunct< \_CharT >, 3179
- ~sentry
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, 3414
- ~set
  - std::set< \_Key, \_Compare, \_Alloc >, 3427
- ~stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3501
- ~temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3578
- ~time\_get
  - std::time\_get< \_CharT, \_InIter >, 3597
- ~time\_put
  - std::time\_put< \_CharT, \_OutIter >, 3625
- ~type\_info
  - std::type\_info, 3686
- ~unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, 3709
  - std::unique\_ptr< \_Tp[], \_Dp >, 3716
- ~vector
  - std::vector< \_Tp, \_Alloc >, 3877
- a
  - std::cauchy\_distribution< \_RealType >, 2173

- std::extreme\_value\_distribution< \_RealType >, [2531](#)
- std::weibull\_distribution< \_RealType >, [3930](#)
- a\_const\_iterator
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3662](#)
- abi, [586](#)
- abs
  - Complex Numbers, [56](#)
- access\_traits
  - \_\_gnu\_pbds::trie< Key, Mapped, \_ATraits, Tag, Node\_Update, \_Alloc >, [3652](#)
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3662](#)
- accumulate
  - Generalized Numeric operations, [106](#), [107](#)
- accumulate\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1279](#)
- acos
  - std, [704](#)
- acosh
  - std, [705](#)
- Adaptors for pointers to functions, [3](#)
  - ptr\_fun, [4](#)
- Adaptors for pointers to members, [5](#)
- add\_const\_t
  - Metaprogramming, [192](#)
- add\_cv\_t
  - Metaprogramming, [192](#)
- add\_lvalue\_reference\_t
  - Metaprogramming, [193](#)
- add\_pointer\_t
  - Metaprogramming, [193](#)
- add\_rvalue\_reference\_t
  - Metaprogramming, [193](#)
- add\_volatile\_t
  - Metaprogramming, [193](#)
- addressof
  - Utilities, [474](#)
- adjacent\_difference
  - Generalized Numeric operations, [108](#)
- adjacent\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1279](#)
- adjacent\_find
  - Non-Mutating, [230](#), [231](#)
- adjustfield
  - std::basic\_fstream< \_CharT, \_Traits >, [1472](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1528](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1561](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1626](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1678](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1732](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1777](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1821](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1865](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2090](#)
- std::ios\_base, [2717](#)
- adopt\_lock
  - Mutexes, [224](#)
- advance
  - std, [705](#)
- airy\_ai
  - Mathematical Special Functions, [151](#)
- airy\_aif
  - Mathematical Special Functions, [151](#)
- airy\_ail
  - Mathematical Special Functions, [152](#)
- airy\_bi
  - Mathematical Special Functions, [152](#)
- airy\_bif
  - Mathematical Special Functions, [152](#)
- airy\_bil
  - Mathematical Special Functions, [152](#)
- algo.h, [3939](#)
- algbase.h, [3948](#)
- algorithm, [3950](#), [3952](#)
- algorithmfwd.h, [3953](#), [3959](#)
- Algorithms, [6](#)
- align
  - Memory, [185](#)
- aligned\_buffer.h, [3967](#)
- aligned\_storage\_t
  - Metaprogramming, [193](#)
- alignment\_value
  - Metaprogramming, [197](#)
- all
  - std::bitset< \_Nb >, [2157](#)
  - std::locale, [2871](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2469](#)
- all\_of
  - Non-Mutating, [231](#)
- alloc\_traits.h, [3967](#), [3968](#)
- allocate
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [871](#)–[873](#)
  - std::allocator\_traits< \_Alloc >, [1309](#), [1310](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1316](#), [1317](#)
- allocate\_shared
  - Pointer Abstractions, [307](#)
- allocated\_ptr.h, [3969](#)
- allocator.h, [3969](#)
- allocator\_type
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3662](#)
  - std::allocator\_traits< \_Alloc >, [1306](#)

- std::allocator\_traits< allocator< \_Tp > >, 1314
- std::set< \_Key, \_Compare, \_Alloc >, 3419
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3723
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3762
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3798
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3833
- Allocators, 7
  - \_\_allocator\_base, 8
  - operator!=, 8
  - operator==, 8
- alpha
  - std::gamma\_distribution< \_RealType >, 2606
- any, 3970
  - std::bitset< \_Nb >, 2157
  - std::experimental::fundamentals\_v1::any, 1322, 1323
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2469
- any\_cast
  - Type-safe container of any type, 460–462
- any\_of
  - Non-Mutating, 232
- app
  - std::basic\_fstream< \_CharT, \_Traits >, 1472
  - std::basic\_ifstream< \_CharT, \_Traits >, 1528
  - std::basic\_ios< \_CharT, \_Traits >, 1561
  - std::basic\_iostream< \_CharT, \_Traits >, 1626
  - std::basic\_istream< \_CharT, \_Traits >, 1679
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1733
  - std::basic\_ofstream< \_CharT, \_Traits >, 1778
  - std::basic\_ostream< \_CharT, \_Traits >, 1821
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1865
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2090
  - std::ios\_base, 2717
- append
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 983–986
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1982, 1983
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1919–1922
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2469, 2470
- apply
  - Numeric Arrays, 269, 270
- apply\_generator
  - \_\_gnu\_cxx::typelist, 515
- arg
  - Complex Numbers, 56
- std, 706
- argument\_type
  - \_\_gnu\_cxx::\_\_detail::\_Ffit\_finder< \_Tp >, 1076
  - \_\_gnu\_cxx::binary\_compose< \_Operation1, \_Operation2, \_Operation3 >, 2118
  - \_\_gnu\_cxx::select1st< \_Pair >, 3409
  - \_\_gnu\_cxx::select2nd< \_Pair >, 3410
  - \_\_gnu\_cxx::subtractive\_rng, 3573
  - \_\_gnu\_cxx::unary\_compose< \_Operation1, \_Operation2 >, 3689
  - \_\_gnu\_parallel::\_binder1st< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 887
  - \_\_gnu\_parallel::\_binder2nd< \_Operation, \_FirstArgumentType, \_SecondArgumentType, \_ResultType >, 889
  - \_\_gnu\_parallel::\_unary\_negate< \_Predicate, argument\_type >, 974
  - std::binder1st< \_Operation >, 2136
  - std::binder2nd< \_Operation >, 2137
  - std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, 2272
  - std::const\_mem\_fun\_t< \_Ret, \_Tp >, 2274
  - std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2640
  - std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2641
  - std::logical\_not< \_Tp >, 2879
  - std::mem\_fun\_ref\_t< \_Ret, \_Tp >, 2955
  - std::mem\_fun\_t< \_Ret, \_Tp >, 2956
  - std::negate< \_Tp >, 3091
  - std::pointer\_to\_unary\_function< \_Arg, \_Result >, 3273
  - std::unary\_function< \_Arg, \_Result >, 3690
  - std::unary\_negate< \_Predicate >, 3691
- Arithmetic Classes, 9
- array, 3971–3973
- Array creation functions, 10
  - make\_array, 10
  - to\_array, 10
- asin
  - std, 706
- asinh
  - std, 706
- assertions.h, 3973
- assign
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 987–991
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1984
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1879–1882
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1923–1927
  - std::deque< \_Tp, \_Alloc >, 2419, 2420
  - std::forward\_list< \_Tp, \_Alloc >, 2557, 2558

- std::list< \_Tp, \_Alloc >, [2838](#), [2839](#)
- std::vector< \_Tp, \_Alloc >, [3877](#), [3878](#)
- assoc\_container.hpp, [3974](#)
- assoc\_laguerre
  - Mathematical Special Functions, [152](#)
  - TR1 Mathematical Special Functions, [442](#)
- assoc\_laguerref
  - Mathematical Special Functions, [154](#)
- assoc\_laguerrel
  - Mathematical Special Functions, [154](#)
- assoc\_legendre
  - Mathematical Special Functions, [154](#)
  - TR1 Mathematical Special Functions, [442](#)
- assoc\_legendref
  - Mathematical Special Functions, [155](#)
- assoc\_legendrel
  - Mathematical Special Functions, [155](#)
- Associative, [11](#)
- async
  - Futures, [104](#)
- at
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [992](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1985](#), [1986](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1927](#), [1928](#)
  - std::deque< \_Tp, \_Alloc >, [2421](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2909](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3730](#)
  - std::vector< \_Tp, \_Alloc >, [3879](#)
- atan
  - std, [706](#)
- atanh
  - std, [706](#)
- ate
  - std::basic\_fstream< \_CharT, \_Traits >, [1472](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1529](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1562](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1626](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1679](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1733](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1778](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1821](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1865](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2091](#)
  - std::ios\_base, [2717](#)
- atomic, [3975](#)
- atomic\_base.h, [3979](#)
- atomic\_bool
  - Atomics, [17](#)
- ATOMIC\_BOOL\_LOCK\_FREE
  - Atomics, [16](#)
- atomic\_char
  - Atomics, [17](#)
- atomic\_char16\_t
  - Atomics, [17](#)
- atomic\_char32\_t
  - Atomics, [17](#)
- atomic\_compare\_exchange\_strong\_explicit
  - Pointer Abstractions, [308](#)
- atomic\_exchange\_explicit
  - Pointer Abstractions, [308](#)
- atomic\_futex.h, [3980](#)
- atomic\_int
  - Atomics, [17](#)
- atomic\_int16\_t
  - Atomics, [17](#)
- atomic\_int32\_t
  - Atomics, [18](#)
- atomic\_int64\_t
  - Atomics, [18](#)
- atomic\_int8\_t
  - Atomics, [18](#)
- atomic\_int\_fast16\_t
  - Atomics, [18](#)
- atomic\_int\_fast32\_t
  - Atomics, [18](#)
- atomic\_int\_fast64\_t
  - Atomics, [19](#)
- atomic\_int\_fast8\_t
  - Atomics, [19](#)
- atomic\_int\_least16\_t
  - Atomics, [19](#)
- atomic\_int\_least32\_t
  - Atomics, [19](#)
- atomic\_int\_least64\_t
  - Atomics, [19](#)
- atomic\_int\_least8\_t
  - Atomics, [20](#)
- atomic\_intmax\_t
  - Atomics, [20](#)
- atomic\_intptr\_t
  - Atomics, [20](#)
- atomic\_is\_lock\_free
  - Pointer Abstractions, [309](#)
- atomic\_llong
  - Atomics, [20](#)
- atomic\_load\_explicit
  - Pointer Abstractions, [309](#)
- atomic\_lockfree\_defines.h, [3980](#)
- atomic\_long
  - Atomics, [20](#)
- atomic\_ptrdiff\_t



Atomics, [21](#)  
 atomic\_schar  
     Atomics, [21](#)  
 atomic\_short  
     Atomics, [21](#)  
 atomic\_size\_t  
     Atomics, [21](#)  
 atomic\_store\_explicit  
     Pointer Abstractions, [310](#)  
 atomic\_uchar  
     Atomics, [21](#)  
 atomic\_uint  
     Atomics, [22](#)  
 atomic\_uint16\_t  
     Atomics, [22](#)  
 atomic\_uint32\_t  
     Atomics, [22](#)  
 atomic\_uint64\_t  
     Atomics, [22](#)  
 atomic\_uint8\_t  
     Atomics, [22](#)  
 atomic\_uint\_fast16\_t  
     Atomics, [23](#)  
 atomic\_uint\_fast32\_t  
     Atomics, [23](#)  
 atomic\_uint\_fast64\_t  
     Atomics, [23](#)  
 atomic\_uint\_fast8\_t  
     Atomics, [23](#)  
 atomic\_uint\_least16\_t  
     Atomics, [23](#)  
 atomic\_uint\_least32\_t  
     Atomics, [24](#)  
 atomic\_uint\_least64\_t  
     Atomics, [24](#)  
 atomic\_uint\_least8\_t  
     Atomics, [24](#)  
 atomic\_uintmax\_t  
     Atomics, [24](#)  
 atomic\_uintptr\_t  
     Atomics, [24](#)  
 atomic\_ullong  
     Atomics, [25](#)  
 atomic\_ulong  
     Atomics, [25](#)  
 atomic\_ushort  
     Atomics, [25](#)  
 atomic\_wchar\_t  
     Atomics, [25](#)  
 atomic\_word.h, [3980](#)  
 atomicity.h, [3981](#)  
 Atomics, [12](#)  
     atomic\_bool, [17](#)  
     ATOMIC\_BOOL\_LOCK\_FREE, [16](#)

atomic\_char, [17](#)  
 atomic\_char16\_t, [17](#)  
 atomic\_char32\_t, [17](#)  
 atomic\_int, [17](#)  
 atomic\_int16\_t, [17](#)  
 atomic\_int32\_t, [18](#)  
 atomic\_int64\_t, [18](#)  
 atomic\_int8\_t, [18](#)  
 atomic\_int\_fast16\_t, [18](#)  
 atomic\_int\_fast32\_t, [18](#)  
 atomic\_int\_fast64\_t, [19](#)  
 atomic\_int\_fast8\_t, [19](#)  
 atomic\_int\_least16\_t, [19](#)  
 atomic\_int\_least32\_t, [19](#)  
 atomic\_int\_least64\_t, [19](#)  
 atomic\_int\_least8\_t, [20](#)  
 atomic\_intmax\_t, [20](#)  
 atomic\_intptr\_t, [20](#)  
 atomic\_llong, [20](#)  
 atomic\_long, [20](#)  
 atomic\_ptrdiff\_t, [21](#)  
 atomic\_schar, [21](#)  
 atomic\_short, [21](#)  
 atomic\_size\_t, [21](#)  
 atomic\_uchar, [21](#)  
 atomic\_uint, [22](#)  
 atomic\_uint16\_t, [22](#)  
 atomic\_uint32\_t, [22](#)  
 atomic\_uint64\_t, [22](#)  
 atomic\_uint8\_t, [22](#)  
 atomic\_uint\_fast16\_t, [23](#)  
 atomic\_uint\_fast32\_t, [23](#)  
 atomic\_uint\_fast64\_t, [23](#)  
 atomic\_uint\_fast8\_t, [23](#)  
 atomic\_uint\_least16\_t, [23](#)  
 atomic\_uint\_least32\_t, [24](#)  
 atomic\_uint\_least64\_t, [24](#)  
 atomic\_uint\_least8\_t, [24](#)  
 atomic\_uintmax\_t, [24](#)  
 atomic\_uintptr\_t, [24](#)  
 atomic\_ullong, [25](#)  
 atomic\_ulong, [25](#)  
 atomic\_ushort, [25](#)  
 atomic\_wchar\_t, [25](#)  
 kill\_dependency, [26](#)  
 memory\_order, [25](#), [26](#)  
 auto\_ptr  
     std::auto\_ptr<\_Tp>, [1362–1364](#)  
 auto\_ptr.h, [3981](#)  
 awk  
     std::regex\_constants, [848](#)  
 b  
     std::extreme\_value\_distribution<\_RealType>, [2531](#)



- std::weibull\_distribution< \_RealType >, 3930
- back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 993
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1986
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1928, 1929
  - std::deque< \_Tp, \_Alloc >, 2422
  - std::list< \_Tp, \_Alloc >, 2840
  - std::queue< \_Tp, \_Sequence >, 3298
  - std::vector< \_Tp, \_Alloc >, 3881
- back\_insert\_iterator
  - std::back\_insert\_iterator< \_Container >, 1370
- back\_inserter
  - Iterators, 137
- backward\_warning.h, 3982
- bad
  - std::basic\_fstream< \_CharT, \_Traits >, 1425
  - std::basic\_ifstream< \_CharT, \_Traits >, 1492
  - std::basic\_ios< \_CharT, \_Traits >, 1546
  - std::basic\_iostream< \_CharT, \_Traits >, 1580
  - std::basic\_istream< \_CharT, \_Traits >, 1643
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1696
  - std::basic\_ofstream< \_CharT, \_Traits >, 1750
  - std::basic\_ostream< \_CharT, \_Traits >, 1795
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1839
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2044
- badbit
  - std::basic\_fstream< \_CharT, \_Traits >, 1472
  - std::basic\_ifstream< \_CharT, \_Traits >, 1529
  - std::basic\_ios< \_CharT, \_Traits >, 1562
  - std::basic\_iostream< \_CharT, \_Traits >, 1626
  - std::basic\_istream< \_CharT, \_Traits >, 1679
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1733
  - std::basic\_ofstream< \_CharT, \_Traits >, 1778
  - std::basic\_ostream< \_CharT, \_Traits >, 1821
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1865
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2091
  - std::ios\_base, 2717
- balanced\_quicksort.h, 3982
- base
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 1226
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1243
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2448
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2688
  - std::reverse\_iterator< \_Iterator >, 3372
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3479
- Base and Implementation Classes, 27, 30
  - \_Opcode, 31
  - \_\_clp2, 29
- Base and Policy Classes, 32–34
- base.h, 3983
- basefield
  - std::basic\_fstream< \_CharT, \_Traits >, 1473
  - std::basic\_ifstream< \_CharT, \_Traits >, 1529
  - std::basic\_ios< \_CharT, \_Traits >, 1562
  - std::basic\_iostream< \_CharT, \_Traits >, 1627
  - std::basic\_istream< \_CharT, \_Traits >, 1679
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1733
  - std::basic\_ofstream< \_CharT, \_Traits >, 1778
  - std::basic\_ostream< \_CharT, \_Traits >, 1821
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1866
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2091
  - std::ios\_base, 2717
- basic
  - std::regex\_constants, 848
- basic\_file.h, 3984
- basic\_filebuf
  - std::basic\_filebuf< \_CharT, \_Traits >, 1389
- basic\_fstream
  - std::basic\_fstream< \_CharT, \_Traits >, 1423
- basic\_ifstream
  - std::basic\_ifstream< \_CharT, \_Traits >, 1491
- basic\_ios
  - std::basic\_ios< \_CharT, \_Traits >, 1545, 1546
- basic\_ios.h, 3984
- basic\_ios.tcc, 3984
- basic\_iostream
  - std::basic\_iostream< \_CharT, \_Traits >, 1579
- basic\_istream
  - std::basic\_istream< \_CharT, \_Traits >, 1643
- basic\_istreamstream
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1695
- basic\_iterator.h, 3985
- basic\_ofstream
  - std::basic\_ofstream< \_CharT, \_Traits >, 1748, 1749
- basic\_ostream
  - std::basic\_ostream< \_CharT, \_Traits >, 1794
- basic\_ostringstream
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1837
- basic\_regex

- std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [1875](#), [1877](#), [1878](#)
- basic\_streambuf
  - std::basic\_streambuf< \_CharT, \_Traits >, [1892](#)
- basic\_string
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1915–1919](#)
- basic\_string.h, [3985](#)
- basic\_string.tcc, [3988](#)
- basic\_stringbuf
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2012](#)
- basic\_stringstream
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2042](#), [2043](#)
- before\_begin
  - std::forward\_list< \_Tp, \_Alloc >, [2559](#)
- beg
  - std::basic\_fstream< \_CharT, \_Traits >, [1473](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1529](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1562](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1627](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1679](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1733](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1778](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1822](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1866](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2091](#)
  - std::ios\_base, [2718](#)
- begin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [993](#), [994](#)
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, [3579](#)
  - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, [1197](#)
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, [3400](#)
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, [3666](#)
- Numeric Arrays, [270](#)
- std, [707](#)
- std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, [1293](#)
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1929](#)
- std::deque< \_Tp, \_Alloc >, [2422](#), [2423](#)
- std::forward\_list< \_Tp, \_Alloc >, [2559](#), [2560](#)
- std::initializer\_list< \_E >, [2693](#)
- std::list< \_Tp, \_Alloc >, [2840](#), [2841](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2910](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2942](#)
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3029](#)
- std::multiset< \_Key, \_Compare, \_Alloc >, [3061](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3427](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3731](#), [3732](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3769](#), [3770](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3804](#), [3805](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3840](#), [3841](#)
- std::vector< \_Tp, \_Alloc >, [3881](#), [3882](#)
- Bernoulli Distributions, [35](#)
- operator!=, [36](#)
- operator<<, [36](#), [38](#)
- operator>>, [38](#), [39](#)
- bernoulli\_distribution
  - std::bernoulli\_distribution, [2099](#)
- beta
  - Mathematical Special Functions, [155](#)
  - std::gamma\_distribution< \_RealType >, [2606](#)
  - TR1 Mathematical Special Functions, [442](#)
- betaf
  - Mathematical Special Functions, [156](#)
- betal
  - Mathematical Special Functions, [156](#)
- bin\_search\_tree.hpp, [3988](#)
- binary
  - std::basic\_fstream< \_CharT, \_Traits >, [1473](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1529](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1562](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1627](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1680](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1734](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1779](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1822](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1866](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2091](#)
  - std::ios\_base, [2718](#)
- Binary Search, [40](#)
  - binary\_search, [41](#)
  - equal\_range, [42](#)
  - lower\_bound, [43](#)
  - upper\_bound, [44](#)
- binary\_heap.hpp, [3989](#)
- binary\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2125](#)
- binary\_heap\_point\_const\_iterator\_
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2130](#), [2131](#)

- binary\_search
  - Binary Search, [41](#)
- bind
  - Binder Classes, [47](#)
- bind1st
  - Binder Classes, [47](#)
- bind2nd
  - Binder Classes, [48](#)
- Binder Classes, [46](#)
  - bind, [47](#)
  - bind1st, [47](#)
  - bind2nd, [48](#)
- binders.h, [3989](#)
- binomial\_heap\_.hpp, [3990](#)
- binomial\_heap\_base\_.hpp, [3990](#)
- bit, [3991](#)
- Bit manipulation, [49](#)
- bitmap\_allocator.h, [3991](#)
  - \_BALLOC\_ALIGN\_BYTES, [3992](#)
- bitset, [3993](#), [3994](#)
  - std::bitset< \_Nb >, [2155](#), [2156](#)
- bool\_set, [3994](#)
  - std::tr2::bool\_set, [2168](#)
- bool\_set.tcc, [3995](#)
- boolalpha
  - std, [709](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1473](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1530](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1563](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1627](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1680](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1734](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1779](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1822](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1866](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2092](#)
  - std::ios\_base, [2718](#)
- Boolean Operations Classes, [50](#)
- boost\_concept\_check.h, [3996](#)
- Branch-Based, [51](#)
- branch\_policy.hpp, [3996](#)
- bucket
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, [1244](#)
- bucket\_count
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3732](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3770](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3806](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3841](#)
- c
  - std::queue< \_Tp, \_Sequence >, [3301](#)
- c++0x\_warning.h, [3997](#)
- c++allocator.h, [3997](#)
- c++config.h, [3997](#)
- c++io.h, [4003](#)
- c++locale.h, [4003](#)
- c++locale\_internal.h, [4004](#)
- c\_str
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [994](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1929](#)
- cache\_line\_size
  - \_\_gnu\_parallel::Settings, [1279](#)
- call\_once
  - Mutexes, [223](#)
  - std::once\_flag, [3194](#)
- capacity
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [994](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1987](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1929](#)
  - std::vector< \_Tp, \_Alloc >, [3882](#)
- cassert, [4004](#)
- cast.h, [4004](#)
- category
  - std::locale, [2863](#)
- cbefore\_begin
  - std::forward\_list< \_Tp, \_Alloc >, [2560](#)
- cbegin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [994](#)
  - std, [709](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1930](#)
  - std::deque< \_Tp, \_Alloc >, [2423](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2560](#)
  - std::list< \_Tp, \_Alloc >, [2841](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2910](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2942](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3029](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3061](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3427](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3732](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3770](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3806](#)

- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3841](#)
- std::vector< \_Tp, \_Alloc >, [3882](#)
- cc\_hash\_max\_collision\_check\_resize\_trigger
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2177](#)
- cc\_hash\_max\_collision\_check\_resize\_trigger\_imp.hpp, [4005](#)
- cc\_hash\_table
  - \_\_gnu\_pbds::cc\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Hash\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, [2183–2186](#)
- cc\_ht\_map.hpp, [4005](#)
- ccomplex, [4006](#)
- cctype, [4006](#), [4007](#)
- cend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [995](#)
- std, [709](#)
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1930](#)
- std::deque< \_Tp, \_Alloc >, [2423](#)
- std::forward\_list< \_Tp, \_Alloc >, [2560](#)
- std::list< \_Tp, \_Alloc >, [2841](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2911](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2942](#)
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3029](#)
- std::multiset< \_Key, \_Compare, \_Alloc >, [3061](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3427](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3733](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3771](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3807](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3842](#)
- std::vector< \_Tp, \_Alloc >, [3882](#)
- cerr
  - std, [793](#)
- cerrno, [4007](#)
- cfenv, [4007](#), [4008](#)
- cfloat, [4008](#)
- char\_traits.h, [4009](#)
- char\_type
  - std::\_\_ctype\_abstract\_base< \_CharT >, [901](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1541](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1891](#)
  - std::collate< \_CharT >, [2243](#)
  - std::collate\_byname< \_CharT >, [2250](#)
  - std::ctype< char >, [2321](#)
  - std::ctype< wchar\_t >, [2338](#)
  - std::ctype\_byname< char >, [2375](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2787](#)
- std::messages< \_CharT >, [2965](#)
- std::money\_get< \_CharT, \_InIter >, [2979](#)
- std::money\_put< \_CharT, \_OutIter >, [2985](#)
- std::moneypunct< \_CharT, \_Intl >, [2992](#)
- std::num\_get< \_CharT, \_InIter >, [3113](#)
- std::num\_put< \_CharT, \_OutIter >, [3135](#)
- std::numpunct< \_CharT >, [3177](#)
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3197](#)
- std::ostreambuf\_iterator< \_CharT, \_Traits >, [3202](#)
- std::time\_get< \_CharT, \_InIter >, [3596](#)
- std::time\_put< \_CharT, \_OutIter >, [3624](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3908](#)
- charconv, [4009](#)
- charconv.h, [4011](#)
- chars\_format
  - std, [690](#)
- checkers.h, [4011](#)
- chrono, [4012](#), [4013](#)
- cin
  - std, [793](#)
- cinttypes, [4014](#)
- ciso646, [4014](#)
- classic
  - std::locale, [2867](#)
- classic\_table
  - std::ctype< char >, [2323](#)
  - std::ctype\_byname< char >, [2375](#)
- clear
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [995](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1425](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1493](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1547](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1580](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1644](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1697](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1751](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1795](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1839](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1930](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2045](#)
  - std::deque< \_Tp, \_Alloc >, [2423](#)
  - std::experimental::fundamentals\_v1::any, [1324](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2561](#)
  - std::list< \_Tp, \_Alloc >, [2841](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2911](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3030](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3062](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3428](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2470](#)

- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3734](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3772](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3807](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3843](#)
- std::vector< \_Tp, \_Alloc >, [3883](#)
- climits, [4015](#)
- locale, [4015](#)
- clog
  - std, [794](#)
- close
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2487](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3501](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1390](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1425](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1493](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1751](#)
- cmath, [4016](#), [4018](#)
- cmp\_fn
  - \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, [3635](#)
- cmp\_fn\_imps.hpp, [4021](#)
- code
  - std::regex\_error, [3345](#)
- codecvt, [4021](#)
- codecvt.h, [4021](#)
- codecvt\_specializations.h, [4022](#)
- collate
  - std::collate< \_CharT >, [2243](#), [2244](#)
  - std::locale, [2871](#)
  - std::regex\_constants, [849](#)
- combine
  - std::locale, [2867](#)
- common\_type\_t
  - Metaprogramming, [194](#)
- comp\_ellint\_1
  - Mathematical Special Functions, [157](#)
  - TR1 Mathematical Special Functions, [443](#)
- comp\_ellint\_1f
  - Mathematical Special Functions, [158](#)
- comp\_ellint\_1l
  - Mathematical Special Functions, [158](#)
- comp\_ellint\_2
  - Mathematical Special Functions, [158](#)
  - TR1 Mathematical Special Functions, [443](#)
- comp\_ellint\_2f
  - Mathematical Special Functions, [159](#)
- comp\_ellint\_2l
  - Mathematical Special Functions, [159](#)
- comp\_ellint\_3
  - Mathematical Special Functions, [159](#)
- TR1 Mathematical Special Functions, [443](#)
- comp\_ellint\_3f
  - Mathematical Special Functions, [160](#)
- comp\_ellint\_3l
  - Mathematical Special Functions, [160](#)
- compare, [4022](#)
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [995–998](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1987](#), [1989](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1930–1933](#)
  - std::collate< \_CharT >, [2244](#)
  - std::collate\_byname< \_CharT >, [2250](#)
  - std::sub\_match< \_Biter >, [3560](#), [3561](#)
- Comparison Classes, [52](#)
- compatibility.h, [4022](#), [4023](#)
- compiletime\_settings.h, [4023](#)
  - \_GLIBCXX\_CALL, [4023](#)
  - \_GLIBCXX\_PARALLEL\_ASSERTIONS, [4024](#)
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_L1, [4024](#)
  - \_GLIBCXX\_RANDOM\_SHUFFLE\_CONSIDER\_TLB, [4024](#)
  - \_GLIBCXX\_SCALE\_DOWN\_FPU, [4024](#)
  - \_GLIBCXX\_VERBOSE\_LEVEL, [4025](#)
- complex, [4025](#), [4030](#)
  - std::complex< \_Tp >, [2259](#)
- Complex Numbers, [53](#)
  - abs, [56](#)
  - arg, [56](#)
  - conj, [57](#)
  - cos, [57](#)
  - cosh, [57](#)
  - exp, [57](#)
  - fabs, [58](#)
  - log, [58](#)
  - log10, [58](#)
  - norm, [58](#)
  - operator!=, [59](#)
  - operator<<, [64](#)
  - operator>>, [66](#)
  - operator\*, [59](#), [60](#)
  - operator\*=, [60](#)
  - operator+, [61](#)
  - operator+=, [62](#)
  - operator-, [62](#), [63](#)
  - operator-=, [63](#)
  - operator/, [63](#), [64](#)
  - operator/=, [64](#)
  - operator=, [65](#)
  - operator==, [65](#), [66](#)
  - polar, [66](#)
  - pow, [66](#), [67](#)

sin, [68](#)  
 sinh, [68](#)  
 sqrt, [68](#)  
 tan, [68](#)  
 tanh, [69](#)  
 complex.h, [4030](#)  
 compose1  
     SGI, [396](#)  
 compose2  
     SGI, [397](#)  
 concept\_check.h, [4031](#)  
 concepts, [4031](#)  
 concurrence.h, [4031](#)  
 Concurrency, [70](#)  
 cond\_dealtor.hpp, [4032](#)  
 cond\_key\_dtor\_entry\_dealtor.hpp, [4032](#)  
 Condition Variables, [71](#)  
     cv\_status, [71](#)  
 condition\_variable, [4033](#)  
 conditional\_t  
     Metaprogramming, [194](#)  
 conf\_hyperg  
     Mathematical Special Functions, [161](#)  
     TR1 Mathematical Special Functions, [443](#)  
 conf\_hypergf  
     Mathematical Special Functions, [161](#)  
 conf\_hypergl  
     Mathematical Special Functions, [162](#)  
 conj  
     Complex Numbers, [57](#)  
 Const-propagating wrapper, [72](#)  
 const\_iterator  
     \_\_gnu\_pbds::trie\_string\_access\_traits< String,  
         Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >,  
         [3665](#)  
     std::set< \_Key, \_Compare, \_Alloc >, [3419](#)  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3723](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3762](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3798](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3833](#)  
 const\_iterator.hpp, [4033](#), [4034](#)  
 const\_local\_iterator  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3723](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3762](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3798](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3834](#)

const\_pointer  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [2123](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [2129](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<  
         Node, \_Alloc >, [2801](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator<  
         Node, \_Alloc >, [2807](#)  
 std::allocator\_traits< \_Alloc >, [1307](#)  
 std::allocator\_traits< allocator< \_Tp > >, [1314](#)  
 std::set< \_Key, \_Compare, \_Alloc >, [3420](#)  
 std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
     loc >, [3724](#)  
 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
     \_Alloc >, [3763](#)  
 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
     loc >, [3799](#)  
 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
     >, [3834](#)  
 const\_pointer\_cast  
     Pointer Abstractions, [310](#)  
     std, [710](#)  
 const\_reference  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator<  
         Node, Const\_Iterator, Iterator, \_Alloc >, [2105](#)  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator<  
         Node, Const\_Iterator, Iterator, \_Alloc >, [2112](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [2124](#)  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator<  
         Value\_Type, Entry, Simple, \_Alloc >, [2129](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator<  
         Node, \_Alloc >, [2801](#)  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator<  
         Node, \_Alloc >, [2807](#)  
     std::set< \_Key, \_Compare, \_Alloc >, [3420](#)  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Al-  
         loc >, [3724](#)  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred,  
         \_Alloc >, [3763](#)  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Al-  
         loc >, [3799](#)  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc  
         >, [3834](#)  
 const\_reverse\_iterator  
     std::set< \_Key, \_Compare, \_Alloc >, [3420](#)  
 const\_void\_pointer  
     \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [870](#)  
     std::allocator\_traits< \_Alloc >, [1307](#)  
     std::allocator\_traits< allocator< \_Tp > >, [1314](#)  
 constant0  
     SGI, [397](#)  
 constant1



- SGL, [397](#)
- constant2
  - SGL, [397](#)
- construct
  - `__gnu_cxx::__alloc_traits< _Alloc, typename >`, [873](#), [874](#)
  - `std::allocator_traits< _Alloc >`, [1310](#)
  - `std::allocator_traits< allocator< _Tp > >`, [1317](#)
- `constructor_destructor_fn_imps.hpp`, [4034](#), [4035](#)
- `constructor_destructor_no_store_hash_fn_imps.hpp`, [4035](#)
- `constructor_destructor_store_hash_fn_imps.hpp`, [4035](#)
- `constructors_destructor_fn_imps.hpp`, [4035–4037](#)
- `container_base_dispatch.hpp`, [4037](#)
- container\_type
  - `std::back_insert_iterator< _Container >`, [1369](#)
  - `std::front_insert_iterator< _Container >`, [2582](#)
  - `std::insert_iterator< _Container >`, [2698](#)
- Containers, [74](#), [75](#)
- converted
  - `std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >`, [3936](#)
- copy
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [999](#)
  - Mutating, [200](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [1934](#)
- copy\_backward
  - Mutating, [200](#)
- copy\_if
  - Mutating, [201](#)
- copy\_n
  - Mutating, [201](#)
  - SGL, [398](#)
- copy\_options
  - Filesystem TS, [99](#)
- copyfmt
  - `std::basic_fstream< _CharT, _Traits >`, [1426](#)
  - `std::basic_ifstream< _CharT, _Traits >`, [1493](#)
  - `std::basic_ios< _CharT, _Traits >`, [1547](#)
  - `std::basic_iostream< _CharT, _Traits >`, [1580](#)
  - `std::basic_istream< _CharT, _Traits >`, [1644](#)
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, [1697](#)
  - `std::basic_ofstream< _CharT, _Traits >`, [1751](#)
  - `std::basic_ostream< _CharT, _Traits >`, [1795](#)
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, [1839](#)
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, [2045](#)
- cos
  - Complex Numbers, [57](#)
- cosh
  - Complex Numbers, [57](#)
- count
  - Non-Mutating, [233](#)
  - `std::bitset< _Nb >`, [2157](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2911](#), [2912](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3030](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3062](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3428](#)
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, [2470](#)
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3734](#)
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, [3772](#)
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, [3808](#)
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, [3843](#)
- count\_if
  - Non-Mutating, [233](#)
- count\_minimal\_n
  - `__gnu_parallel::Settings`, [1279](#)
- cout
  - std, [794](#)
- cpp\_type\_traits.h, [4038](#)
- cpu\_defines.h, [4039](#)
- crbegin
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [1000](#)
  - std, [710](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [1935](#)
  - `std::deque< _Tp, _Alloc >`, [2424](#)
  - `std::list< _Tp, _Alloc >`, [2842](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2912](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3031](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3063](#)
  - `std::set< _Key, _Compare, _Alloc >`, [3429](#)
  - `std::vector< _Tp, _Alloc >`, [3883](#)
- cref
  - `std::reference_wrapper< _Tp >`, [3342](#)
- cregex\_token\_iterator
  - Regular Expressions, [349](#)
- crend
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, [1000](#)
  - std, [710](#)
  - `std::basic_string< _CharT, _Traits, _Alloc >`, [1935](#)
  - `std::deque< _Tp, _Alloc >`, [2424](#)
  - `std::list< _Tp, _Alloc >`, [2842](#)
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, [2912](#)
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, [3031](#)
  - `std::multiset< _Key, _Compare, _Alloc >`, [3063](#)

- std::set< \_Key, \_Compare, \_Alloc >, 3429
- std::vector< \_Tp, \_Alloc >, 3883
- csetjmp, 4039
- cshift
  - Numeric Arrays, 271
- csignal, 4039
- cstdalign, 4040
- cstdarg, 4040
- cstdbool, 4041
- cstddef, 4041
- cstdint, 4042
- cstdio, 4042, 4043
- cstdlib, 4043, 4044
- cstring, 4044
- csub\_match
  - Regular Expressions, 349
- ctgmach, 4044, 4045
- ctime, 4045
- ctype
  - std::ctype< char >, 2322
  - std::ctype< wchar\_t >, 2338
  - std::locale, 2872
- ctype\_base.h, 4046
- ctype\_inline.h, 4046
- cuchar, 4046
- cur
  - std::basic\_fstream< \_CharT, \_Traits >, 1474
  - std::basic\_ifstream< \_CharT, \_Traits >, 1530
  - std::basic\_ios< \_CharT, \_Traits >, 1563
  - std::basic\_iostream< \_CharT, \_Traits >, 1628
  - std::basic\_istream< \_CharT, \_Traits >, 1680
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1734
  - std::basic\_ofstream< \_CharT, \_Traits >, 1779
  - std::basic\_ostream< \_CharT, \_Traits >, 1822
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1867
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2092
  - std::ios\_base, 2718
- curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, 2994
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3005
- current\_exception
  - Exceptions, 89
- cv\_status
  - Condition Variables, 71
- cwchar, 4047
- cwctype, 4048
- cxxabi.h, 4048
  - \_\_cxa\_demangle, 4050
- cxxabi\_forced.h, 4051
- cxxabi\_init\_exception.h, 4051
- cxxabi\_tweaks.h, 4051
- cyl\_bessel\_i
  - Mathematical Special Functions, 162
  - TR1 Mathematical Special Functions, 444
- cyl\_bessel\_if
  - Mathematical Special Functions, 163
- cyl\_bessel\_il
  - Mathematical Special Functions, 163
- cyl\_bessel\_j
  - Mathematical Special Functions, 163
  - TR1 Mathematical Special Functions, 444
- cyl\_bessel\_jf
  - Mathematical Special Functions, 164
- cyl\_bessel\_jl
  - Mathematical Special Functions, 164
- cyl\_bessel\_k
  - Mathematical Special Functions, 164
  - TR1 Mathematical Special Functions, 444
- cyl\_bessel\_kf
  - Mathematical Special Functions, 165
- cyl\_bessel\_kl
  - Mathematical Special Functions, 165
- cyl\_neumann
  - Mathematical Special Functions, 166
  - TR1 Mathematical Special Functions, 444
- cyl\_neumannf
  - Mathematical Special Functions, 166
- cyl\_neumannl
  - Mathematical Special Functions, 167
- data
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1000
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1935
  - std::vector< \_Tp, \_Alloc >, 3883
- Data Structure Type, 76
- date\_order
  - std::time\_get< \_CharT, \_InIter >, 3597
  - std::time\_get\_byname< \_CharT, \_InIter >, 3611
- deallocate
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 874, 875
  - std::allocator\_traits< \_Alloc >, 1311
  - std::allocator\_traits< allocator< \_Tp > >, 1318
- debug.h, 4052
- debug\_allocator.h, 4053
- debug\_fn\_imps.hpp, 4053–4055
- debug\_map\_base.hpp, 4055
- debug\_no\_store\_hash\_fn\_imps.hpp, 4056
- debug\_store\_hash\_fn\_imps.hpp, 4056
- dec
  - std, 711
  - std::basic\_fstream< \_CharT, \_Traits >, 1474
  - std::basic\_ifstream< \_CharT, \_Traits >, 1530
  - std::basic\_ios< \_CharT, \_Traits >, 1563



- std::basic\_iostream< \_CharT, \_Traits >, 1628
- std::basic\_istream< \_CharT, \_Traits >, 1680
- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1734
- std::basic\_ofstream< \_CharT, \_Traits >, 1779
- std::basic\_ostream< \_CharT, \_Traits >, 1823
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1867
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2092
- std::ios\_base, 2719
- decay\_t
  - Metaprogramming, 194
- decimal, 4056
- Decimal Floating-Point Arithmetic, 77
- decimal128
  - std::decimal::decimal128, 2391
- decimal32
  - std::decimal::decimal32, 2393
- decimal32\_to\_long\_long
  - std::decimal, 834
- decimal64
  - std::decimal::decimal64, 2395
- decimal\_point
  - std::moneypunct< \_CharT, \_Intl >, 2994
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3006
  - std::numpunct< \_CharT >, 3179
  - std::numpunct\_byname< \_CharT >, 3186
- declare\_no\_pointers
  - Pointer Safety and Garbage Collection, 324
- declare\_reachable
  - Pointer Safety and Garbage Collection, 324
- default\_delete
  - std::default\_delete< \_Tp >, 2396, 2397
  - std::default\_delete< \_Tp[] >, 2398
- defaultfloat
  - std, 711
- defer\_lock
  - Mutexes, 225
- denorm\_absent
  - std, 691
- denorm\_indeterminate
  - std, 691
- denorm\_min
  - std::numeric\_limits< \_Tp >, 3151
- denorm\_present
  - std, 691
- densities
  - std::piecewise\_constant\_distribution< \_RealType >, 3260
  - std::piecewise\_linear\_distribution< \_RealType >, 3265
- deque, 4066, 4067
  - std::deque< \_Tp, \_Alloc >, 2411–2414
- deque.tcc, 4067
- destroy
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, 875, 876
  - std::allocator\_traits< \_Alloc >, 1311
  - std::allocator\_traits< allocator< \_Tp > >, 1318
- Diagnostics, 78
  - generic\_category, 80
  - make\_error\_code, 80
  - make\_error\_condition, 80
  - operator!=, 80
  - operator<, 81
  - operator==, 81
  - system\_category, 81
- difference\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 2105
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 2112
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 2124
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 2129
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 2801
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 2807
  - std::allocator\_traits< \_Alloc >, 1307
  - std::allocator\_traits< allocator< \_Tp > >, 1314
  - std::back\_insert\_iterator< \_Container >, 1369
  - std::front\_insert\_iterator< \_Container >, 2582
  - std::insert\_iterator< \_Container >, 2698
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2783
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2787
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2794
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3198
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 3203
  - std::pointer\_traits< \_Ptr >, 3275
  - std::pointer\_traits< \_Tp \* >, 3276
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 3322
  - std::set< \_Key, \_Compare, \_Alloc >, 3420
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3724
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3763
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3799
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3834
- digits
  - std::\_\_numeric\_limits\_base, 949

- std::numeric\_limits< \_Tp >, [3153](#)
- digits10
  - std::\_\_numeric\_limits\_base, [950](#)
  - std::numeric\_limits< \_Tp >, [3153](#)
- direct\_mask\_range\_hashing\_imp.hpp, [4069](#)
- direct\_mod\_range\_hashing\_imp.hpp, [4069](#)
- discard
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2448](#)
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2688](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2820](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2960](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3479](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3568](#)
- discard\_block\_engine
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2446](#), [2447](#)
- distance
  - SGL, [398](#)
  - std, [711](#)
- do\_compare
  - std::collate< \_CharT >, [2245](#)
  - std::collate\_byname< \_CharT >, [2251](#)
- do\_curr\_symbol
  - std::moneypunct< \_CharT, \_Intl >, [2994](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3006](#)
- do\_date\_order
  - std::time\_get< \_CharT, \_InIter >, [3597](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3611](#)
- do\_decimal\_point
  - std::moneypunct< \_CharT, \_Intl >, [2995](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3006](#)
  - std::numpunct< \_CharT >, [3180](#)
  - std::numpunct\_byname< \_CharT >, [3186](#)
- do\_falsename
  - std::numpunct< \_CharT >, [3180](#)
  - std::numpunct\_byname< \_CharT >, [3186](#)
- do\_frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, [2995](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3007](#)
- do\_get
  - std::messages< \_CharT >, [2966](#)
  - std::messages\_byname< \_CharT >, [2969](#)
  - std::money\_get< \_CharT, \_InIter >, [2980](#)
  - std::num\_get< \_CharT, \_InIter >, [3114](#)–[3121](#)
  - std::time\_get< \_CharT, \_InIter >, [3598](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3611](#)
- do\_get\_date
  - std::time\_get< \_CharT, \_InIter >, [3599](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3612](#)
- do\_get\_monthname
  - std::time\_get< \_CharT, \_InIter >, [3599](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3613](#)
- do\_get\_time
  - std::time\_get< \_CharT, \_InIter >, [3600](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3614](#)
- do\_get\_weekday
  - std::time\_get< \_CharT, \_InIter >, [3601](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3615](#)
- do\_get\_year
  - std::time\_get< \_CharT, \_InIter >, [3602](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3615](#)
- do\_grouping
  - std::moneypunct< \_CharT, \_Intl >, [2996](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3007](#)
  - std::numpunct< \_CharT >, [3180](#)
  - std::numpunct\_byname< \_CharT >, [3187](#)
- do\_hash
  - std::collate< \_CharT >, [2245](#)
  - std::collate\_byname< \_CharT >, [2251](#)
- do\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, [901](#)
  - std::ctype< \_CharT >, [2303](#), [2304](#)
  - std::ctype< wchar\_t >, [2339](#), [2340](#)
  - std::ctype\_byname< \_CharT >, [2357](#), [2358](#)
- do\_narrow
  - std::\_\_ctype\_abstract\_base< \_CharT >, [902](#), [903](#)
  - std::ctype< \_CharT >, [2304](#), [2305](#)
  - std::ctype< char >, [2323](#)
  - std::ctype< wchar\_t >, [2340](#), [2341](#)
  - std::ctype\_byname< \_CharT >, [2358](#), [2359](#)
  - std::ctype\_byname< char >, [2375](#), [2376](#)
- do\_neg\_format
  - std::moneypunct< \_CharT, \_Intl >, [2996](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3008](#)
- do\_negative\_sign
  - std::moneypunct< \_CharT, \_Intl >, [2997](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3008](#)
- do\_out
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, [891](#)
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, [2206](#)
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, [2211](#)
  - std::codecvt< char, char, mbstate\_t >, [2216](#)
  - std::codecvt< char16\_t, char, mbstate\_t >, [2221](#)
  - std::codecvt< char32\_t, char, mbstate\_t >, [2226](#)
  - std::codecvt< wchar\_t, char, mbstate\_t >, [2231](#)
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, [2237](#)
- do\_pos\_format
  - std::moneypunct< \_CharT, \_Intl >, [2997](#)

- std::moneypunct\_byname< \_CharT, \_Intl >, 3009
- do\_positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, 2998
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3009
- do\_put
  - std::money\_put< \_CharT, \_Outlter >, 2986, 2987
  - std::num\_put< \_CharT, \_Outlter >, 3136–3140
  - std::time\_put< \_CharT, \_Outlter >, 3626
  - std::time\_put\_byname< \_CharT, \_Outlter >, 3630
- do\_scan\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 903
  - std::ctype< \_CharT >, 2306
  - std::ctype< wchar\_t >, 2341
  - std::ctype\_byname< \_CharT >, 2359
- do\_scan\_not
  - std::\_\_ctype\_abstract\_base< \_CharT >, 904
  - std::ctype< \_CharT >, 2306
  - std::ctype< wchar\_t >, 2342
  - std::ctype\_byname< \_CharT >, 2360
- do\_thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 2998
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3010
  - std::numput< \_CharT >, 3181
  - std::numput\_byname< \_CharT >, 3187
- do\_tolower
  - std::\_\_ctype\_abstract\_base< \_CharT >, 905
  - std::ctype< \_CharT >, 2307, 2308
  - std::ctype< char >, 2324, 2325
  - std::ctype< wchar\_t >, 2343
  - std::ctype\_byname< \_CharT >, 2361
  - std::ctype\_byname< char >, 2376, 2377
- do\_toupper
  - std::\_\_ctype\_abstract\_base< \_CharT >, 906
  - std::ctype< \_CharT >, 2308, 2309
  - std::ctype< char >, 2325, 2326
  - std::ctype< wchar\_t >, 2344
  - std::ctype\_byname< \_CharT >, 2362
  - std::ctype\_byname< char >, 2377, 2378
- do\_transform
  - std::collate< \_CharT >, 2246
  - std::collate\_byname< \_CharT >, 2252
- do\_truename
  - std::numput< \_CharT >, 3181
  - std::numput\_byname< \_CharT >, 3188
- do\_widen
  - std::\_\_ctype\_abstract\_base< \_CharT >, 907, 908
  - std::ctype< \_CharT >, 2309, 2310
  - std::ctype< char >, 2326, 2327
  - std::ctype< wchar\_t >, 2346
  - std::ctype\_byname< \_CharT >, 2363, 2364
  - std::ctype\_byname< char >, 2378, 2379
- duration\_cast
  - Time, 454
- Dynamic Bitset., 82
- operator!=, 83
- operator<<, 84
- operator<=, 84
- operator>, 84
- operator>>, 85
- operator>=, 85
- operator^, 85
- operator-, 84
- operator&, 83
- operator|, 86
- dynamic\_bitset, 4069
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2466–2468
- dynamic\_bitset.tcc, 4070
- dynamic\_pointer\_cast
  - Pointer Abstractions, 310
  - std, 712
- e\_pos
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 3400
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3666
- e\_type
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, 3400
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String, Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3665
- eback
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2487
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3502
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3527
  - std::basic\_filebuf< \_CharT, \_Traits >, 1390
  - std::basic\_streambuf< \_CharT, \_Traits >, 1893
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2013
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3910
- ECMAScript
  - std::regex\_constants, 849
- egptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2488
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3502
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3527
  - std::basic\_filebuf< \_CharT, \_Traits >, 1391
  - std::basic\_streambuf< \_CharT, \_Traits >, 1893
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2013
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3911
- egrep
  - std::regex\_constants, 849
- element\_type
  - std::auto\_ptr< \_Tp >, 1362
  - std::pointer\_traits< \_Ptr >, 3275
  - std::pointer\_traits< \_Tp \* >, 3276

- std::shared\_ptr< \_Tp >, [3466](#)
- ellint\_1
  - Mathematical Special Functions, [167](#)
  - TR1 Mathematical Special Functions, [445](#)
- ellint\_1f
  - Mathematical Special Functions, [168](#)
- ellint\_1l
  - Mathematical Special Functions, [168](#)
- ellint\_2
  - Mathematical Special Functions, [168](#)
  - TR1 Mathematical Special Functions, [445](#)
- ellint\_2f
  - Mathematical Special Functions, [169](#)
- ellint\_2l
  - Mathematical Special Functions, [170](#)
- ellint\_3
  - Mathematical Special Functions, [170](#)
  - TR1 Mathematical Special Functions, [445](#)
- ellint\_3f
  - Mathematical Special Functions, [171](#)
- ellint\_3l
  - Mathematical Special Functions, [171](#)
- emplace
  - std::deque< \_Tp, \_Alloc >, [2424](#)
  - std::list< \_Tp, \_Alloc >, [2842](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2913](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3031](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3063](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3429](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3734](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3772](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3808](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3843](#)
  - std::vector< \_Tp, \_Alloc >, [3884](#)
- emplace\_after
  - std::forward\_list< \_Tp, \_Alloc >, [2561](#)
- emplace\_front
  - std::forward\_list< \_Tp, \_Alloc >, [2562](#)
- emplace\_hint
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2913](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3032](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3064](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3430](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3735](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3773](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3809](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3844](#)
- empty
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1001](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1990](#)
  - \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [2191](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2623](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1935](#)
  - std::deque< \_Tp, \_Alloc >, [2425](#)
  - std::experimental::fundamentals\_v1::any, [1324](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2562](#)
  - std::list< \_Tp, \_Alloc >, [2843](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2914](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2943](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3032](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3064](#)
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3286](#)
  - std::queue< \_Tp, \_Sequence >, [3299](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3430](#)
  - std::stack< \_Tp, \_Sequence >, [3495](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2470](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3736](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3773](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3809](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3845](#)
  - std::vector< \_Tp, \_Alloc >, [3884](#)
- enable\_if\_t
  - Metaprogramming, [194](#)
- enable\_special\_members.h, [4071](#)
- enc\_filebuf.h, [4071](#)
- end
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1001](#)
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, [3579](#)
  - \_\_gnu\_parallel::PseudoSequence< \_Tp, \_DifferenceTp >, [1197](#)
  - \_\_gnu\_pbds::sample\_trie\_access\_traits, [3400](#)
  - \_\_gnu\_pbds::trie\_string\_access\_traits< String,

- Min\_E\_Val, Max\_E\_Val, Reverse, \_Alloc >, 3666
- Numeric Arrays, 271, 272
- std, 712, 713
- std::Temporary\_buffer< \_ForwardIterator, \_Tp >, 1293
- std::basic\_fstream< \_CharT, \_Traits >, 1474
- std::basic\_ifstream< \_CharT, \_Traits >, 1530
- std::basic\_ios< \_CharT, \_Traits >, 1563
- std::basic\_iostream< \_CharT, \_Traits >, 1628
- std::basic\_istream< \_CharT, \_Traits >, 1681
- std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1735
- std::basic\_ofstream< \_CharT, \_Traits >, 1780
- std::basic\_ostream< \_CharT, \_Traits >, 1823
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1867
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1936
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2092
- std::deque< \_Tp, \_Alloc >, 2425
- std::forward\_list< \_Tp, \_Alloc >, 2562
- std::initializer\_list< \_E >, 2694
- std::ios\_base, 2719
- std::list< \_Tp, \_Alloc >, 2843
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2914
- std::match\_results< \_Bi\_iter, \_Alloc >, 2943
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3033
- std::multiset< \_Key, \_Compare, \_Alloc >, 3065
- std::set< \_Key, \_Compare, \_Alloc >, 3431
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3736, 3737
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3774, 3775
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3809, 3810
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3845, 3846
- std::vector< \_Tp, \_Alloc >, 3884, 3885
- endl
- std, 713
- ends
- std, 713
- entry\_cmp.hpp, 4071
- entry\_list\_fn\_imps.hpp, 4072
- entry\_metadata\_base.hpp, 4072
- entry\_pred.hpp, 4072
- eof
  - std::basic\_fstream< \_CharT, \_Traits >, 1426
  - std::basic\_ifstream< \_CharT, \_Traits >, 1494
  - std::basic\_ios< \_CharT, \_Traits >, 1548
  - std::basic\_iostream< \_CharT, \_Traits >, 1581
  - std::basic\_istream< \_CharT, \_Traits >, 1645
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1698
  - std::basic\_ofstream< \_CharT, \_Traits >, 1752
  - std::basic\_ostream< \_CharT, \_Traits >, 1796
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1840
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2046
- eofbit
  - std::basic\_fstream< \_CharT, \_Traits >, 1474
  - std::basic\_ifstream< \_CharT, \_Traits >, 1531
  - std::basic\_ios< \_CharT, \_Traits >, 1564
  - std::basic\_iostream< \_CharT, \_Traits >, 1628
  - std::basic\_istream< \_CharT, \_Traits >, 1681
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1735
  - std::basic\_ofstream< \_CharT, \_Traits >, 1780
  - std::basic\_ostream< \_CharT, \_Traits >, 1823
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1867
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2093
  - std::ios\_base, 2719
- ep\_ptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2488
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3502
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3528
  - std::basic\_filebuf< \_CharT, \_Traits >, 1391
  - std::basic\_streambuf< \_CharT, \_Traits >, 1893
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2013
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3911
- epsilon
  - std::numeric\_limits< \_Tp >, 3151
- eq\_by\_less.hpp, 4073
- equal
  - Non-Mutating, 234, 235
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2790
- equal\_range
  - Binary Search, 42
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2915, 2916
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3033–3035
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3065–3067
  - std::set< \_Key, \_Compare, \_Alloc >, 3431–3433
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3737, 3738
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3775, 3776
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3811
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3846, 3847

- equally\_split.h, 4073
- equals
  - std::tr2::bool\_set, 2169
- erase
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1002, 1003
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1990
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1936, 1937
  - std::deque< \_Tp, \_Alloc >, 2425, 2426
  - std::list< \_Tp, \_Alloc >, 2843, 2844
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2917–2919
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3035–3037
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3067, 3068
  - std::set< \_Key, \_Compare, \_Alloc >, 3433, 3434
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3738–3740
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3776–3778
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3812, 3814
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3847–3849
  - std::vector< \_Tp, \_Alloc >, 3885, 3886
- erase\_after
  - std::forward\_list< \_Tp, \_Alloc >, 2563
- erase\_can\_throw
  - \_\_gnu\_pbds::container\_traits< Cntnr >, 2294
- erase\_fn\_imps.hpp, 4073–4076
- erase\_if.h, 4076
- erase\_no\_store\_hash\_fn\_imps.hpp, 4076
- erase\_store\_hash\_fn\_imps.hpp, 4076, 4077
- error\_backref
  - std::regex\_constants, 842
- error\_badbrace
  - std::regex\_constants, 842
- error\_badrepeat
  - std::regex\_constants, 842
- error\_brace
  - std::regex\_constants, 842
- error\_brack
  - std::regex\_constants, 842
- error\_collate
  - std::regex\_constants, 842
- error\_complexity
  - std::regex\_constants, 842
- error\_constants.h, 4077
- error\_ctype
  - std::regex\_constants, 843
- error\_escape
  - std::regex\_constants, 843
- error\_paren
  - std::regex\_constants, 843
- error\_range
  - std::regex\_constants, 843
- error\_space
  - std::regex\_constants, 843
- error\_stack
  - std::regex\_constants, 843
- error\_type
  - std::regex\_constants, 841
- event
  - std::basic\_fstream< \_CharT, \_Traits >, 1422
  - std::basic\_ifstream< \_CharT, \_Traits >, 1490
  - std::basic\_ios< \_CharT, \_Traits >, 1545
  - std::basic\_iostream< \_CharT, \_Traits >, 1578
  - std::basic\_istream< \_CharT, \_Traits >, 1642
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1694
  - std::basic\_ofstream< \_CharT, \_Traits >, 1748
  - std::basic\_ostream< \_CharT, \_Traits >, 1793
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1836
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2042
  - std::ios\_base, 2709
- event\_callback
  - std::basic\_fstream< \_CharT, \_Traits >, 1420
  - std::basic\_ifstream< \_CharT, \_Traits >, 1488
  - std::basic\_ios< \_CharT, \_Traits >, 1541
  - std::basic\_iostream< \_CharT, \_Traits >, 1576
  - std::basic\_istream< \_CharT, \_Traits >, 1640
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1692
  - std::basic\_ofstream< \_CharT, \_Traits >, 1746
  - std::basic\_ostream< \_CharT, \_Traits >, 1791
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1834
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2039
  - std::ios\_base, 2707
- exception, 4077
- exception.h, 4078
- exception.hpp, 4079
- exception\_defines.h, 4079
- exception\_ptr.h, 4079
- Exceptions, 87, 92
  - \_\_verbose\_terminate\_handler, 89
  - current\_exception, 89
  - get\_terminate, 89
  - get\_unexpected, 89
  - make\_exception\_ptr, 90
  - rethrow\_exception, 90
  - rethrow\_if\_nested, 90
  - set\_terminate, 90



- set\_unexpected, [90](#)
- terminate, [91](#)
- terminate\_handler, [88](#)
- throw\_with\_nested, [91](#)
- uncaught\_exception, [91](#)
- uncaught\_exceptions, [91](#)
- unexpected, [91](#)
- unexpected\_handler, [89](#)
- what, [91](#)
- exceptions
  - std::basic\_fstream< \_CharT, \_Traits >, [1426](#), [1427](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1494](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1548](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1581](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1645](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1698](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1752](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1796](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1840](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2046](#)
- exchange
  - std, [714](#)
- exp
  - Complex Numbers, [57](#)
- experimental/algorithm
  - sample, [3953](#)
- experimental/bits/shared\_ptr.h
  - get\_deleter, [4265](#)
- experimental/functional
  - is\_bind\_expression\_v, [4103](#)
  - is\_placeholder\_v, [4103](#)
  - make\_boyer\_moore\_horspool\_searcher, [4102](#)
  - make\_boyer\_moore\_searcher, [4102](#)
  - make\_default\_searcher, [4102](#)
  - not\_fn, [4102](#)
- experimental/iterator
  - make\_ostream\_joiner, [4133](#)
- experimental/memory\_resource
  - get\_default\_resource, [4161](#)
  - set\_default\_resource, [4161](#)
- experimental/numeric
  - gcd, [4185](#)
  - lcm, [4185](#)
- expint
  - Mathematical Special Functions, [171](#)
  - TR1 Mathematical Special Functions, [445](#)
- expintf
  - Mathematical Special Functions, [172](#)
- expintl
  - Mathematical Special Functions, [172](#)
- exponential\_distribution
  - std::exponential\_distribution< \_RealType >, [2525](#), [2526](#)
- extc++.h, [4080](#)
- extended
  - std::regex\_constants, [849](#)
- Extensions, [93](#)
- external\_load\_access
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2177](#)
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2666](#)
- extptr\_allocator.h, [4080](#)
- fabs
  - Complex Numbers, [58](#)
  - std, [714](#)
- facet
  - std::locale::facet, [2535](#)
- fail
  - std::basic\_fstream< \_CharT, \_Traits >, [1427](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1495](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1549](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1582](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1646](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1699](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1753](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1797](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1841](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2047](#)
- failbit
  - std::basic\_fstream< \_CharT, \_Traits >, [1475](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1531](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1564](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1629](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1681](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1735](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1780](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1823](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1868](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2093](#)
  - std::ios\_base, [2719](#)
- failed
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3205](#)
- false\_type
  - Metaprogramming, [194](#)
- falsename
  - std::numpunct< \_CharT >, [3182](#)
  - std::numpunct\_byname< \_CharT >, [3188](#)

- fd
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3503
- features.h, 4081
  - `_GLIBCXX_BAL_QUICKSORT`, 4081
  - `_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS`, 4081
  - `_GLIBCXX_FIND_EQUAL_SPLIT`, 4081
  - `_GLIBCXX_FIND_GROWING_BLOCKS`, 4082
  - `_GLIBCXX_MERGESORT`, 4082
  - `_GLIBCXX_QUICKSORT`, 4082
  - `_GLIBCXX_TREE_DYNAMIC_BALANCING`, 4082
  - `_GLIBCXX_TREE_FULL_COPY`, 4083
  - `_GLIBCXX_TREE_INITIAL_SPLITTING`, 4083
- fenv.h, 4083
- file
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3503
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3528
- filebuf
  - I/O, 126
- filesystem, 4084
- Filesystem TS, 94
  - copy\_options, 99
  - perms, 99
- fill
  - Mutating, 202
  - `std::basic_fstream< _CharT, _Traits >`, 1428
  - `std::basic_ifstream< _CharT, _Traits >`, 1495, 1496
  - `std::basic_ios< _CharT, _Traits >`, 1549
  - `std::basic_iostream< _CharT, _Traits >`, 1582, 1583
  - `std::basic_istream< _CharT, _Traits >`, 1646
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1699
  - `std::basic_ofstream< _CharT, _Traits >`, 1753
  - `std::basic_ostream< _CharT, _Traits >`, 1797, 1798
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1841, 1842
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2047
- fill\_minimal\_n
  - `__gnu_parallel::Settings`, 1279
- fill\_n
  - Mutating, 203
- find
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1003–1005
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1991
  - Non-Mutating, 236
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1938, 1939
  - `std::map< _Key, _Tp, _Compare, _Alloc >`, 2919–2921
  - `std::multimap< _Key, _Tp, _Compare, _Alloc >`, 3038, 3039
  - `std::multiset< _Key, _Compare, _Alloc >`, 3069, 3070
  - `std::set< _Key, _Compare, _Alloc >`, 3435, 3436
  - `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3740, 3742
  - `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3778, 3779
  - `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3816
  - `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3849, 3851
- find.h, 4084
- find\_by\_order
  - `__gnu_pbds::tree_order_statistics_node_update< Node_Cltr, Node_Itr, Cmp_Fn, _Alloc >`, 3640, 3641
  - `__gnu_pbds::trie_order_statistics_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3657
- find\_end
  - Non-Mutating, 236, 237
- find\_first
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2471
- find\_first\_not\_of
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1006–1008
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1991
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1940–1942
- find\_first\_of
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1008–1010
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1992
  - Non-Mutating, 238
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1942–1944
- find\_fn\_imps.hpp, 4085, 4086
- find\_if
  - Non-Mutating, 239
- find\_if\_not
  - Non-Mutating, 240
- find\_increasing\_factor
  - `__gnu_parallel::Settings`, 1280
- find\_initial\_block\_size
  - `__gnu_parallel::Settings`, 1280
- find\_last\_not\_of
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1010–1012
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1992
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1944–1946



- find\_last\_of
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1013](#), [1014](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1993](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1946–1948](#)
- find\_maximum\_block\_size
  - \_\_gnu\_parallel::Settings, [1280](#)
- find\_next
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2471](#)
- find\_no\_store\_hash\_fn\_imps.hpp, [4086](#)
- find\_scale\_factor
  - \_\_gnu\_parallel::Settings, [1280](#)
- find\_selectors.h, [4087](#)
- find\_sequential\_search\_size
  - \_\_gnu\_parallel::Settings, [1280](#)
- find\_store\_hash\_fn\_imps.hpp, [4087](#)
- first
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, [1123](#)
  - std::pair< \_T1, \_T2 >, [3230](#)
  - std::sub\_match< \_Bilter >, [3566](#)
- first\_argument\_type
  - \_\_gnu\_cxx::project1st< \_Arg1, \_Arg2 >, [3290](#)
  - \_\_gnu\_cxx::project2nd< \_Arg1, \_Arg2 >, [3291](#)
  - \_\_gnu\_parallel::EqualFromLess< \_T1, \_T2, \_Compare >, [1069](#)
  - \_\_gnu\_parallel::EqualTo< \_T1, \_T2 >, [1072](#)
  - \_\_gnu\_parallel::Less< \_T1, \_T2 >, [1128](#)
  - \_\_gnu\_parallel::Lexicographic< \_T1, \_T2, \_Compare >, [1130](#)
  - \_\_gnu\_parallel::LexicographicReverse< \_T1, \_T2, \_Compare >, [1131](#)
  - \_\_gnu\_parallel::Multiplies< \_Tp1, \_Tp2, \_Result >, [1171](#)
  - \_\_gnu\_parallel::Plus< \_Tp1, \_Tp2, \_Result >, [1188](#)
  - std::binary\_function< \_Arg1, \_Arg2, \_Result >, [2119](#)
  - std::binary\_negate< \_Predicate >, [2134](#)
  - std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, [2269](#)
  - std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, [2271](#)
  - std::divides< \_Tp >, [2457](#)
  - std::equal\_to< \_Tp >, [2516](#)
  - std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, [3219](#)
  - std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, [3224](#)
  - std::greater< \_Tp >, [2627](#)
  - std::greater\_equal< \_Tp >, [2629](#)
  - std::less< \_Tp >, [2812](#)
  - std::less\_equal< \_Tp >, [2814](#)
  - std::logical\_and< \_Tp >, [2876](#)
  - std::logical\_or< \_Tp >, [2880](#)
  - std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, [2952](#)
  - std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, [2953](#)
  - std::minus< \_Tp >, [2971](#)
  - std::modulus< \_Tp >, [2974](#)
  - std::multiplies< \_Tp >, [3052](#)
  - std::not\_equal\_to< \_Tp >, [3107](#)
  - std::owner\_less< shared\_ptr< \_Tp > >, [3220](#)
  - std::owner\_less< void >, [3221](#)
  - std::owner\_less< weak\_ptr< \_Tp > >, [3223](#)
  - std::plus< \_Tp >, [3269](#)
  - std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, [3272](#)
- first\_type
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, [1120](#)
  - std::pair< \_T1, \_T2 >, [3228](#)
  - std::sub\_match< \_Bilter >, [3560](#)
- fixed
  - std, [714](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1475](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1531](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1564](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1629](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1681](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1735](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1780](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1824](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1868](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2093](#)
  - std::ios\_base, [2720](#)
- flags
  - std::basic\_fstream< \_CharT, \_Traits >, [1429](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1496](#), [1497](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1550](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1583](#), [1584](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1647](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1700](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1754](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1798](#), [1799](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1842](#), [1843](#)
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [1883](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2048](#)
  - std::ios\_base, [2710](#)
- flip
  - std::bitset< \_Nb >, [2157](#), [2158](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2472](#)
- float\_denorm\_style
  - std, [690](#)

- float\_round\_style
  - std, [691](#)
- floatfield
  - std::basic\_fstream< \_CharT, \_Traits >, [1475](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1531](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1564](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1629](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1682](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1736](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1781](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1824](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1868](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2093](#)
  - std::ios\_base, [2720](#)
- flush
  - std, [714](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1429](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1584](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1755](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1799](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1843](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2049](#)
- fmtflags
  - std::basic\_fstream< \_CharT, \_Traits >, [1420](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1488](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1542](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1576](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1640](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1692](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1746](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1791](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1834](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2040](#)
  - std::ios\_base, [2707](#)
- for\_each
  - Non-Mutating, [240](#)
- for\_each.h, [4087](#)
- for\_each\_minimal\_n
  - \_\_gnu\_parallel:: Settings, [1281](#)
- for\_each\_selectors.h, [4088](#)
- format
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2943](#), [2944](#)
- format\_default
  - std::regex\_constants, [850](#)
- format\_first\_only
  - std::regex\_constants, [850](#)
- format\_no\_copy
  - std::regex\_constants, [850](#)
- format\_sed
  - std::regex\_constants, [851](#)
- formatter.h, [4088](#)
- forward
  - Utilities, [475](#)
- forward\_as\_tuple
  - Utilities, [475](#)
- forward\_list, [4089–4091](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2553–2557](#)
- forward\_list.h, [4091](#)
- forward\_list.tcc, [4092](#)
- fpos
  - std::fpos< \_StateT >, [2577](#)
- frac\_digits
  - std::moneypunct< \_CharT, \_Intl >, [2999](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3010](#)
- from\_bytes
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, [3936](#), [3937](#)
- from\_chars
  - std, [715](#)
- front
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1015](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1993](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1948](#), [1949](#)
  - std::deque< \_Tp, \_Alloc >, [2426](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2564](#)
  - std::list< \_Tp, \_Alloc >, [2844](#), [2845](#)
  - std::queue< \_Tp, \_Sequence >, [3299](#)
  - std::vector< \_Tp, \_Alloc >, [3886](#)
- front\_insert\_iterator
  - std::front\_insert\_iterator< \_Container >, [2584](#)
- front\_inserter
  - Iterators, [138](#)
- fs\_dir.h, [4093](#)
- fs\_fwd.h, [4093](#), [4094](#)
- fs\_path.h, [4095](#), [4096](#)
- fstream, [4096](#)
  - I/O, [126](#)
- fstream.tcc, [4097](#)
- functexcept.h, [4097](#)
- function
  - std::function< \_Res(\_ArgTypes...) >, [2587](#), [2588](#)
- Function Objects, [100](#)
  - mem\_fn, [101](#)
- functional, [4098](#), [4100](#), [4101](#)
- functional\_hash.h, [4103](#)
- functions.h, [4104](#)
- future, [4106](#)

- std::future< \_Res >, 2596
- std::future< \_Res & >, 2599
- std::future< void >, 2602
- future\_category
  - Futures, 104
- future\_errc
  - Futures, 103
- future\_status
  - Futures, 103
- Futures, 102
  - async, 104
  - future\_category, 104
  - future\_errc, 103
  - future\_status, 103
  - launch, 104
  - make\_error\_code, 105
  - make\_error\_condition, 105
  - swap, 105
- gamma\_distribution
  - std::gamma\_distribution< \_RealType >, 2605
- gbump
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2488
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3503
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3528
  - std::basic\_filebuf< \_CharT, \_Traits >, 1391
  - std::basic\_streambuf< \_CharT, \_Traits >, 1894
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2014
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3911
- gcd
  - experimental/numeric, 4185
- gcount
  - std::basic\_fstream< \_CharT, \_Traits >, 1430
  - std::basic\_ifstream< \_CharT, \_Traits >, 1497
  - std::basic\_iostream< \_CharT, \_Traits >, 1584
  - std::basic\_istream< \_CharT, \_Traits >, 1648
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1701
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2049
- Generalized Numeric operations, 106
  - accumulate, 106, 107
  - adjacent\_difference, 108
  - inner\_product, 109
  - iota, 110
  - partial\_sum, 111
- generate
  - Mutating, 203
- generate\_canonical
  - Random Number Generation, 334
- generate\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1281
- generate\_n
  - Mutating, 204
- generic\_category
  - Diagnostics, 80
- get
  - \_\_gnu\_parallel::Settings, 1278
  - std::\_\_allocated\_ptr< \_Alloc >, 879
  - std::auto\_ptr< \_Tp >, 1364
  - std::basic\_fstream< \_CharT, \_Traits >, 1430–1433
  - std::basic\_ifstream< \_CharT, \_Traits >, 1497–1500
  - std::basic\_iostream< \_CharT, \_Traits >, 1585–1588
  - std::basic\_istream< \_CharT, \_Traits >, 1648–1651
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1701–1704
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2049–2052
  - std::future< \_Res >, 2596
  - std::future< \_Res & >, 2599
  - std::future< void >, 2602
  - std::money\_get< \_CharT, \_InIter >, 2981, 2982
  - std::num\_get< \_CharT, \_InIter >, 3122, 3124–3131
  - std::shared\_future< \_Res >, 3454
  - std::shared\_future< \_Res & >, 3458
  - std::shared\_ptr< \_Tp >, 3473
  - std::time\_get< \_CharT, \_InIter >, 3602, 3603
  - std::time\_get\_byname< \_CharT, \_InIter >, 3616, 3617
  - std::unique\_ptr< \_Tp, \_Dp >, 3709
  - std::unique\_ptr< \_Tp[], \_Dp >, 3716
  - Utilities, 476, 477
- get\_actual\_size
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2680
- get\_allocator
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1015
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1994
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1949
  - std::deque< \_Tp, \_Alloc >, 2427
  - std::forward\_list< \_Tp, \_Alloc >, 2564
  - std::list< \_Tp, \_Alloc >, 2845
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2921
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2944
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3040
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3071
  - std::set< \_Key, \_Compare, \_Alloc >, 3437
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2472
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3742
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3779

- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3817
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3851
- `std::vector< _Tp, _Alloc >`, 3887
- `get_child`
  - `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1175
  - `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1180
- `get_comb_hash_fn`
  - `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`, 2191
- `get_comb_probe_fn`
  - `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`, 2624
- `get_date`
  - `std::time_get< _CharT, _InIter >`, 3604
  - `std::time_get_byname< _CharT, _InIter >`, 3618
- `get_default_resource`
  - `experimental/memory_resource`, 4161
- `get_deleter`
  - `experimental/bits/shared_ptr.h`, 4265
  - Pointer Abstractions, 311
  - `std::unique_ptr< _Tp, _Dp >`, 3709, 3710
  - `std::unique_ptr< _Tp[], _Dp >`, 3716, 3717
- `get_eq_fn`
  - `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`, 2191
  - `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`, 2624
- `get_hash_fn`
  - `__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >`, 2192
  - `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`, 2625
- `get_id`
  - `std::this_thread`, 856
- `get_l_child`
  - `__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2107
  - `__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2113
- `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >`, 3213
- `__gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >`, 3215
- `get_load`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`, 2177
- `get_loads`
  - `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`, 2667
- `get_metadata`
  - `__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2107
  - `__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2113
  - `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1175
  - `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1180
- `get_money`
  - `std`, 715
- `get_monthname`
  - `std::time_get< _CharT, _InIter >`, 3605
  - `std::time_get_byname< _CharT, _InIter >`, 3618
- `get_nearest_larger_size`
  - `__gnu_pbds::sample_size_policy`, 3398
- `get_nearest_smaller_size`
  - `__gnu_pbds::sample_size_policy`, 3398
- `get_new_handler`
  - `std`, 715
- `get_new_size`
  - `__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >`, 2680
  - `__gnu_pbds::sample_resize_policy`, 3390
- `get_pointer_safety`
  - Pointer Safety and Garbage Collection, 324
- `get_probe_fn`
  - `__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >`, 2625
- `get_r_child`
  - `__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2107
  - `__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2114
  - `__gnu_pbds::detail::ov_tree_node_const_it< Value_Type, Metadata_Type, _Alloc >`, 3213
  - `__gnu_pbds::detail::ov_tree_node_it< Value_Type, Metadata_Type, _Alloc >`, 3215
- `get_resize_policy`

- \_\_gnu\_pbds::detail::cc\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Hash\_Fn, Resize\_Policy >, [2192](#)
  - \_\_gnu\_pbds::detail::gp\_ht\_map< Key, Mapped, Hash\_Fn, Eq\_Fn, \_Alloc, Store\_Hash, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy >, [2626](#)
- get\_size\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [2680](#)
- get\_temporary\_buffer
  - std, [716](#)
- get\_terminate
  - Exceptions, [89](#)
- get\_time
  - std, [716](#)
  - std::time\_get< \_CharT, \_InIter >, [3606](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3619](#)
- get\_trigger\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, [2681](#)
- get\_unexpected
  - Exceptions, [89](#)
- get\_weekday
  - std::time\_get< \_CharT, \_InIter >, [3606](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3620](#)
- get\_year
  - std::time\_get< \_CharT, \_InIter >, [3607](#)
  - std::time\_get\_byname< \_CharT, \_InIter >, [3621](#)
- getline
  - std, [717–719](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1433–1435](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1501](#), [1502](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1588](#), [1589](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1651–1653](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1704–1706](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2053](#), [2054](#)
- getloc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2489](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3504](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3529](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1392](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1435](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1502](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1551](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1590](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1653](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1706](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [1755](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1799](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1843](#)
- std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [1883](#)
- std::basic\_streambuf< \_CharT, \_Traits >, [1894](#)
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2014](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2054](#)
- std::ios\_base, [2711](#)
- std::regex\_traits< \_Ch\_type >, [3356](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3912](#)
- global
  - std::locale, [2867](#)
- good
  - std::basic\_fstream< \_CharT, \_Traits >, [1435](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1502](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1551](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1590](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1653](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1706](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1755](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1800](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1844](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2054](#)
- goodbit
  - std::basic\_fstream< \_CharT, \_Traits >, [1475](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1532](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1565](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1629](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1682](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1736](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1781](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1824](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1868](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2094](#)
  - std::ios\_base, [2720](#)
- gp\_hash\_table
  - \_\_gnu\_pbds::gp\_hash\_table< Key, Mapped, Hash\_Fn, Eq\_Fn, Comb\_Probe\_Fn, Probe\_Fn, Resize\_Policy, Store\_Hash, \_Alloc >, [2615–2619](#)
- gp\_ht\_map.hpp, [4107](#)
- gptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2489](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3504](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3529](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1392](#)

- std::basic\_streambuf< \_CharT, \_Traits >, 1894
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2014
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3912
- grep
  - std::regex\_constants, 851
- grouping
  - std::moneypunct< \_CharT, \_Intl >, 2999
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3011
  - std::numpunct< \_CharT >, 3182
  - std::numpunct\_byname< \_CharT >, 3188
- gslice
  - Numeric Arrays, 264, 265
- gslice.h, 4108
- gslice\_array
  - Numeric Arrays, 265
- gslice\_array.h, 4108
- has\_denorm
  - std::\_\_numeric\_limits\_base, 950
  - std::numeric\_limits< \_Tp >, 3153
- has\_denorm\_loss
  - std::\_\_numeric\_limits\_base, 950
  - std::numeric\_limits< \_Tp >, 3153
- has\_facet
  - Locales, 145
  - std::locale, 2870
  - std::locale::id, 2683
- has\_infinity
  - std::\_\_numeric\_limits\_base, 950
  - std::numeric\_limits< \_Tp >, 3153
- has\_quiet\_NaN
  - std::\_\_numeric\_limits\_base, 950
  - std::numeric\_limits< \_Tp >, 3154
- has\_signaling\_NaN
  - std::\_\_numeric\_limits\_base, 951
  - std::numeric\_limits< \_Tp >, 3154
- hash
  - std::collate< \_CharT >, 2247
  - std::collate\_byname< \_CharT >, 2253
- Hash-Based, 113
- hash\_bytes.h, 4109
- hash\_eq\_fn.hpp, 4109
- hash\_exponential\_size\_policy
  - \_\_gnu\_pbds::hash\_exponential\_size\_policy< Size\_Type >, 2664
- hash\_exponential\_size\_policy\_imp.hpp, 4110
- hash\_fun.h, 4110
- hash\_function
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3742
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3779
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3817
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3851
- hash\_load\_check\_resize\_trigger
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2667
- hash\_load\_check\_resize\_trigger\_imp.hpp, 4110
- hash\_load\_check\_resize\_trigger\_size\_base.hpp, 4110
- hash\_map, 4111
- hash\_policy.hpp, 4112
- hash\_prime\_size\_policy
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 2675
- hash\_prime\_size\_policy\_imp.hpp, 4113
- hash\_set, 4113
- hash\_standard\_resize\_policy
  - \_\_gnu\_pbds::hash\_standard\_resize\_policy< Size\_Policy, Trigger\_Policy, External\_Size\_Access, Size\_Type >, 2679
- hash\_standard\_resize\_policy\_imp.hpp, 4114
- hasher
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3724
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3763
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3799
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3835
- Hashes, 114
- hashtable.h, 4114
- hashtable\_policy.h, 4115
- Heap, 115
  - is\_heap, 115, 116
  - is\_heap\_until, 116, 117
  - make\_heap, 117, 118
  - pop\_heap, 118, 119
  - push\_heap, 119, 120
  - sort\_heap, 120, 121
- Heap-Based, 122
  - priority\_queue, 123
- helper\_functions.h, 4117
- hermite
  - Mathematical Special Functions, 172
  - TR1 Mathematical Special Functions, 446
- hermitef
  - Mathematical Special Functions, 173
- hermitel
  - Mathematical Special Functions, 173
- hex
  - std, 719
  - std::basic\_fstream< \_CharT, \_Traits >, 1476
  - std::basic\_ifstream< \_CharT, \_Traits >, 1532
  - std::basic\_ios< \_CharT, \_Traits >, 1565
  - std::basic\_iostream< \_CharT, \_Traits >, 1630
  - std::basic\_istream< \_CharT, \_Traits >, 1682



- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1736
- std::basic\_ofstream< \_CharT, \_Traits >, 1781
- std::basic\_ostream< \_CharT, \_Traits >, 1824
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1869
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2094
- std::ios\_base, 2720
- hexfloat
  - std, 720
- high\_resolution\_clock
  - Time, 453
- hours
  - Time, 453
- hyperg
  - Mathematical Special Functions, 174
  - TR1 Mathematical Special Functions, 446
- hypergf
  - Mathematical Special Functions, 174
- hypergl
  - Mathematical Special Functions, 175
- I/O, 125
  - filebuf, 126
  - fstream, 126
  - ifstream, 127
  - ios, 127
  - iostream, 127
  - istream, 127
  - istreamstream, 127
  - ofstream, 128
  - ostream, 128
  - ostringstream, 128
  - streambuf, 128
  - stringbuf, 128
  - stringstream, 129
  - wfilebuf, 129
  - wfstream, 129
  - wifstream, 129
  - wios, 129
  - wiostream, 130
  - wistream, 130
  - wistreamstream, 130
  - wofstream, 130
  - wostream, 130
  - wostreamstream, 131
  - wstreambuf, 131
  - wstringbuf, 131
  - wstringstream, 131
- icase
  - std::regex\_constants, 851
- id
  - std::collate< \_CharT >, 2248
- std::collate\_byname< \_CharT >, 2254
- std::ctype< \_CharT >, 2318
- std::ctype< char >, 2335
- std::ctype< wchar\_t >, 2354
- std::ctype\_byname< \_CharT >, 2372
- std::ctype\_byname< char >, 2388
- std::locale::id, 2683
- std::messages< \_CharT >, 2966
- std::messages\_byname< \_CharT >, 2970
- std::money\_get< \_CharT, \_InIter >, 2983
- std::money\_put< \_CharT, \_OutIter >, 2989
- std::moneypunct< \_CharT, \_Intl >, 3002
- std::moneypunct\_byname< \_CharT, \_Intl >, 3014
- std::num\_get< \_CharT, \_InIter >, 3132
- std::num\_put< \_CharT, \_OutIter >, 3149
- std::numpunct< \_CharT >, 3184
- std::numpunct\_byname< \_CharT >, 3190
- std::time\_get< \_CharT, \_InIter >, 3608
- std::time\_get\_byname< \_CharT, \_InIter >, 3621
- std::time\_put< \_CharT, \_OutIter >, 3628
- std::time\_put\_byname< \_CharT, \_OutIter >, 3632
- identity\_element
  - SGI, 398, 399
- ifstream
  - I/O, 127
- ignore
  - std::basic\_fstream< \_CharT, \_Traits >, 1435, 1436
  - std::basic\_ifstream< \_CharT, \_Traits >, 1503, 1504
  - std::basic\_iostream< \_CharT, \_Traits >, 1590, 1591
  - std::basic\_istream< \_CharT, \_Traits >, 1653, 1654
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1706, 1707
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2055, 2056
- imbue
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2489
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3504
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3529
  - std::basic\_filebuf< \_CharT, \_Traits >, 1392
  - std::basic\_fstream< \_CharT, \_Traits >, 1437
  - std::basic\_ifstream< \_CharT, \_Traits >, 1504
  - std::basic\_ios< \_CharT, \_Traits >, 1551
  - std::basic\_iostream< \_CharT, \_Traits >, 1592
  - std::basic\_istream< \_CharT, \_Traits >, 1655
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1708
  - std::basic\_ofstream< \_CharT, \_Traits >, 1756
  - std::basic\_ostream< \_CharT, \_Traits >, 1800
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1844
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1883
  - std::basic\_streambuf< \_CharT, \_Traits >, 1895
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2015

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2056
- std::ios\_base, 2711
- std::regex\_traits< \_Ch\_type >, 3356
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3912
- in
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 891
  - std::basic\_fstream< \_CharT, \_Traits >, 1476
  - std::basic\_ifstream< \_CharT, \_Traits >, 1532
  - std::basic\_ios< \_CharT, \_Traits >, 1565
  - std::basic\_iostream< \_CharT, \_Traits >, 1630
  - std::basic\_istream< \_CharT, \_Traits >, 1682
  - std::basic\_istringstream< \_CharT, \_Traits, \_Alloc >, 1736
  - std::basic\_ofstream< \_CharT, \_Traits >, 1781
  - std::basic\_ostream< \_CharT, \_Traits >, 1825
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1869
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2094
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 2206
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2212
  - std::codecvt< char, char, mbstate\_t >, 2217
  - std::codecvt< char16\_t, char, mbstate\_t >, 2222
  - std::codecvt< char32\_t, char, mbstate\_t >, 2227
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2232
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2238
  - std::ios\_base, 2721
- in\_avail
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2490
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3505
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3530
  - std::basic\_filebuf< \_CharT, \_Traits >, 1393
  - std::basic\_streambuf< \_CharT, \_Traits >, 1895
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2015
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3913
- in\_place
  - Optional values, 303
- includes
  - Set Operations, 405
- increment
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2823
- independent\_bits\_engine
  - std::independent\_bits\_engine< \_\_RandomNumberEngine, \_\_w, \_UIntType >, 2686, 2687
- index\_sequence
  - std, 688
- index\_sequence\_for
  - std, 688
- indirect\_array
  - Numeric Arrays, 265
- indirect\_array.h, 4118
- infinity
  - std::numeric\_limits< \_Tp >, 3151
- info\_fn\_imps.hpp, 4119, 4120
- init
  - std::basic\_fstream< \_CharT, \_Traits >, 1437
  - std::basic\_ifstream< \_CharT, \_Traits >, 1505
  - std::basic\_ios< \_CharT, \_Traits >, 1552
  - std::basic\_iostream< \_CharT, \_Traits >, 1592
  - std::basic\_istream< \_CharT, \_Traits >, 1655
  - std::basic\_istringstream< \_CharT, \_Traits, \_Alloc >, 1708
  - std::basic\_ofstream< \_CharT, \_Traits >, 1756
  - std::basic\_ostream< \_CharT, \_Traits >, 1801
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1845
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2057
- initializer\_list, 4120
- inner\_product
  - Generalized Numeric operations, 109
- inplace\_merge
  - Sorting, 415, 416
- insert
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1016–1021
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1994–1997
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1949–1954
  - std::deque< \_Tp, \_Alloc >, 2427–2429
  - std::list< \_Tp, \_Alloc >, 2845–2847
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2921–2926
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3040–3044
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3071–3073
  - std::set< \_Key, \_Compare, \_Alloc >, 3437–3439
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3743–3747
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3780–3784
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3817–3820
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3852–3855
  - std::vector< \_Tp, \_Alloc >, 3887–3889
- insert\_after
  - std::forward\_list< \_Tp, \_Alloc >, 2564–2566
- insert\_fn\_imps.hpp, 4121–4123
- insert\_iterator
  - std::insert\_iterator< \_Container >, 2699



- insert\_join\_fn\_imps.hpp, [4123](#)
- insert\_no\_store\_hash\_fn\_imps.hpp, [4123](#)
- insert\_store\_hash\_fn\_imps.hpp, [4123](#)
- inserter
  - Iterators, [138](#)
- int\_type
  - std::basic\_ios< \_CharT, \_Traits >, [1542](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1891](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2787](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3909](#)
- internal
  - std, [720](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1476](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1532](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1565](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1630](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1683](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1737](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1782](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1825](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1869](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2094](#)
  - std::ios\_base, [2721](#)
- intervals
  - std::piecewise\_constant\_distribution< \_RealType >, [3260](#)
  - std::piecewise\_linear\_distribution< \_RealType >, [3265](#)
- intl
  - std::moneypunct< \_CharT, \_Intl >, [3003](#)
- Invalidation Guarantees, [132](#)
- invoke.h, [4123](#)
- io\_errc
  - std, [691](#)
- iomanip, [4124](#)
- ios, [4126](#)
  - I/O, [127](#)
- ios\_base.h, [4126](#)
- iosfwd, [4128](#)
- iostate
  - std::basic\_fstream< \_CharT, \_Traits >, [1421](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1489](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1543](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1577](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1641](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1693](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1747](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1792](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1835](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2040](#)
  - std::ios\_base, [2708](#)
- iostream, [4129](#)
  - I/O, [127](#)
- iota
  - Generalized Numeric operations, [110](#)
- is
  - std::\_\_ctype\_abstract\_base< \_CharT >, [908](#), [909](#)
  - std::ctype< \_CharT >, [2311](#)
  - std::ctype< char >, [2327](#), [2328](#)
  - std::ctype< wchar\_t >, [2347](#), [2348](#)
  - std::ctype\_byname< \_CharT >, [2364](#), [2365](#)
  - std::ctype\_byname< char >, [2380](#)
- is\_always\_equal
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [870](#)
  - std::allocator\_traits< \_Alloc >, [1307](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1315](#)
- is\_bind\_expression\_v
  - experimental/functional, [4103](#)
- is\_bounded
  - std::\_\_numeric\_limits\_base, [951](#)
  - std::numeric\_limits< \_Tp >, [3154](#)
- is\_emptyset
  - std::tr2::bool\_set, [2169](#)
- is\_exact
  - std::\_\_numeric\_limits\_base, [951](#)
  - std::numeric\_limits< \_Tp >, [3154](#)
- is\_grow\_needed
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2177](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [3394](#)
- is\_heap
  - Heap, [115](#), [116](#)
- is\_heap\_until
  - Heap, [116](#), [117](#)
- is\_iec559
  - std::\_\_numeric\_limits\_base, [951](#)
  - std::numeric\_limits< \_Tp >, [3154](#)
- is\_indeterminate
  - std::tr2::bool\_set, [2169](#)
- is\_integer
  - std::\_\_numeric\_limits\_base, [951](#)
  - std::numeric\_limits< \_Tp >, [3155](#)
- is\_modulo
  - std::\_\_numeric\_limits\_base, [952](#)
  - std::numeric\_limits< \_Tp >, [3155](#)
- is\_nothrow\_swappable\_v
  - std, [794](#)
- is\_nothrow\_swappable\_with\_v
  - std, [794](#)
- is\_open
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2490](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3505](#)

std::basic\_filebuf< \_CharT, \_Traits >, 1393  
 std::basic\_fstream< \_CharT, \_Traits >, 1438  
 std::basic\_ifstream< \_CharT, \_Traits >, 1505  
 std::basic\_ofstream< \_CharT, \_Traits >, 1757  
 is\_partitioned  
     Mutating, 204  
 is\_permutation  
     Non-Mutating, 241, 243  
 is\_placeholder\_v  
     experimental/functional, 4103  
 is\_resize\_needed  
     \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger<I/O, 127  
         External\_Load\_Access, Size\_Type >, 2177  
     \_\_gnu\_pbds::sample\_resize\_policy, 3390  
     \_\_gnu\_pbds::sample\_resize\_trigger, 3394  
 is\_signed  
     std::\_\_numeric\_limits\_base, 952  
     std::numeric\_limits< \_Tp >, 3155  
 is\_singleton  
     std::tr2::bool\_set, 2169  
 is\_sorted  
     Sorting, 416, 417  
 is\_sorted\_until  
     Sorting, 417, 418  
 is\_specialized  
     std::\_\_numeric\_limits\_base, 952  
     std::numeric\_limits< \_Tp >, 3155  
 is\_swappable\_v  
     std, 794  
 is\_swappable\_with\_v  
     std, 795  
 isalnum  
     std, 720  
 isalpha  
     std, 720  
 isblank  
     std, 721  
 iscntrl  
     std, 721  
 isctype  
     std::regex\_traits< \_Ch\_type >, 3357  
 isdigit  
     std, 721  
 isgraph  
     std, 721  
 islower  
     std, 722  
 isprint  
     std, 722  
 ispunct  
     std, 722  
 isspace  
     std, 722  
 istream, 4130

I/O, 127  
 istream.tcc, 4131  
 istream\_iterator  
     std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2784  
 istream\_type  
     std::istreambuf\_iterator< \_CharT, \_Traits >, 2788  
 istreambuf\_iterator  
     std::istreambuf\_iterator< \_CharT, \_Traits >, 2789, 2790  
 istringstream  
 isupper  
     std, 723  
 isxdigit  
     std, 723  
 iter\_swap  
     Mutating, 205  
 iter\_type  
     std::money\_get< \_CharT, \_InIter >, 2979  
     std::money\_put< \_CharT, \_OutIter >, 2985  
     std::num\_get< \_CharT, \_InIter >, 3113  
     std::num\_put< \_CharT, \_OutIter >, 3135  
     std::time\_get< \_CharT, \_InIter >, 3596  
     std::time\_put< \_CharT, \_OutIter >, 3625  
 iterator, 4132, 4133  
     std::set< \_Key, \_Compare, \_Alloc >, 3421  
     std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3725  
     std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3764  
     std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3800  
     std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3835  
 Iterator Tags, 133  
 iterator.h, 4134  
 iterator.hpp, 4134  
 iterator\_category  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 2106  
     \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it\_< Node, Const\_Iterator, Iterator, \_Alloc >, 2112  
     \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 2124  
     \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator\_< Value\_Type, Entry, Simple, \_Alloc >, 2129  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_< Node, \_Alloc >, 2801  
     \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_< Node, \_Alloc >, 2807  
     std::back\_insert\_iterator< \_Container >, 1369  
     std::front\_insert\_iterator< \_Container >, 2583  
     std::insert\_iterator< \_Container >, 2698

- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2783
- std::istreambuf\_iterator< \_CharT, \_Traits >, 2788
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2794
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3198
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3203
- std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 3322
- std::reverse\_iterator< \_Iterator >, 3371
- iterator\_concepts.h, 4134
- iterator\_fn\_imps.hpp, 4135
- Iterators, 134
  - \_\_iterator\_category, 137
  - back\_inserter, 137
  - front\_inserter, 138
  - inserter, 138
  - make\_reverse\_iterator, 139
  - operator==, 139
- iterators\_fn\_imps.hpp, 4135, 4136
- iword
  - std::basic\_fstream< \_CharT, \_Traits >, 1438
  - std::basic\_ifstream< \_CharT, \_Traits >, 1505
  - std::basic\_ios< \_CharT, \_Traits >, 1552
  - std::basic\_iostream< \_CharT, \_Traits >, 1593
  - std::basic\_istream< \_CharT, \_Traits >, 1656
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1709
  - std::basic\_ofstream< \_CharT, \_Traits >, 1757
  - std::basic\_ostream< \_CharT, \_Traits >, 1801
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1845
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2057
  - std::ios\_base, 2712
- k
  - std::negative\_binomial\_distribution< \_IntType >, 3094
- key\_comp
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2926
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3044
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3073
  - std::set< \_Key, \_Compare, \_Alloc >, 3439
- key\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, 3421
- key\_eq
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3747
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3784
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3820
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3855
- key\_equal
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3725
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3764
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3800
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3835
- key\_type
  - std::set< \_Key, \_Compare, \_Alloc >, 3421
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3725
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3764
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3800
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3835
- kill\_dependency
  - Atomics, 26
- L1\_cache\_size
  - \_\_gnu\_parallel::Settings, 1281
- L2\_cache\_size
  - \_\_gnu\_parallel::Settings, 1281
- laguerre
  - Mathematical Special Functions, 175
  - TR1 Mathematical Special Functions, 446
- laguerref
  - Mathematical Special Functions, 176
- laguerrel
  - Mathematical Special Functions, 176
- lambda
  - std::exponential\_distribution< \_RealType >, 2526
- launch
  - Futures, 104
- lcm
  - experimental/numeric, 4185
- left
  - std, 723
  - std::basic\_fstream< \_CharT, \_Traits >, 1477
  - std::basic\_ifstream< \_CharT, \_Traits >, 1533
  - std::basic\_ios< \_CharT, \_Traits >, 1566
  - std::basic\_iostream< \_CharT, \_Traits >, 1631
  - std::basic\_istream< \_CharT, \_Traits >, 1683
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1737
  - std::basic\_ofstream< \_CharT, \_Traits >, 1782
  - std::basic\_ostream< \_CharT, \_Traits >, 1825
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1870

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2095
- std::ios\_base, 2721
- left\_child\_next\_sibling\_heap.hpp, 4136
- left\_child\_next\_sibling\_heap\_const\_iterator\_
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator\_
    - Node, \_Alloc >, 2802, 2803
- left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator\_
    - Node, \_Alloc >, 2808, 2809
- legendre
  - Mathematical Special Functions, 176
  - TR1 Mathematical Special Functions, 446
- legendref
  - Mathematical Special Functions, 177
- legendrel
  - Mathematical Special Functions, 177
- length
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1022
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1997
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1955
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2945
  - std::regex\_traits< \_Ch\_type >, 3357
  - std::sub\_match< \_BiIter >, 3562
- lexicographical\_compare
  - Sorting, 418, 419
- lexicographical\_compare\_3way
  - SGI, 399
- lfts\_config.h, 4136
- Library Fundamentals TS, 141
- limits, 4137
- linear\_congruential\_engine
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2819
- linear\_probe\_fn\_imp.hpp, 4138
- list, 4138, 4139
  - std::list< \_Tp, \_Alloc >, 2835–2837
- List-Based, 143
- list.tcc, 4140
- list\_partition
  - \_\_gnu\_parallel, 570
- list\_partition.h, 4140
- list\_update
  - \_\_gnu\_pbds::list\_update< Key, Mapped, Eq\_Fn, Update\_Policy, \_Alloc >, 2860
- list\_update\_policy.hpp, 4141
- load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3748
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3784
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3821
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3855
- local\_iterator
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3725
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3764
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3800
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3836
- locale, 4141
  - std::locale, 2863–2866
- locale\_classes.h, 4142
- locale\_classes.tcc, 4142
- locale\_conv.h, 4143
- locale\_facets.h, 4143
- locale\_facets.tcc, 4145
- locale\_facets\_nonio.h, 4146
- locale\_facets\_nonio.tcc, 4146
- localefwd.h, 4147
- Locales, 144
  - has\_facet, 145
  - use\_facet, 145
- lock
  - Mutexes, 223
- log
  - Complex Numbers, 58
- log10
  - Complex Numbers, 58
- logic\_error
  - std::logic\_error, 2875
- lookup\_classname
  - std::regex\_traits< \_Ch\_type >, 3358
- lookup\_collatename
  - std::regex\_traits< \_Ch\_type >, 3359
- losertree.h, 4148
- lower\_bound
  - Binary Search, 43
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2926–2928
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3045, 3046
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3073–3075
  - std::set< \_Key, \_Compare, \_Alloc >, 3439–3441
- lowest
  - std::numeric\_limits< \_Tp >, 3151
- lu\_counter\_metadata.hpp, 4149
- lu\_map.hpp, 4149
- m
  - std::fisher\_f\_distribution< \_RealType >, 2540

- std::lognormal\_distribution< \_RealType >, 2883
- macros.h, 4150
  - \_\_GLIBCXX\_DEBUG\_VERIFY\_COND\_AT, 4154
  - \_\_glibcxx\_check\_erase, 4151
  - \_\_glibcxx\_check\_erase\_after, 4151
  - \_\_glibcxx\_check\_erase\_range, 4151
  - \_\_glibcxx\_check\_erase\_range\_after, 4151
  - \_\_glibcxx\_check\_heap\_pred, 4151
  - \_\_glibcxx\_check\_insert, 4152
  - \_\_glibcxx\_check\_insert\_after, 4152
  - \_\_glibcxx\_check\_insert\_range, 4152
  - \_\_glibcxx\_check\_insert\_range\_after, 4152
  - \_\_glibcxx\_check\_partitioned\_lower, 4153
  - \_\_glibcxx\_check\_partitioned\_lower\_pred, 4153
  - \_\_glibcxx\_check\_partitioned\_upper\_pred, 4153
  - \_\_glibcxx\_check\_sorted\_pred, 4153
- make\_array
  - Array creation functions, 10
- make\_boyer\_moore\_horspool\_searcher
  - experimental/functional, 4102
- make\_boyer\_moore\_searcher
  - experimental/functional, 4102
- make\_default\_searcher
  - experimental/functional, 4102
- make\_error\_code
  - Diagnostics, 80
  - Futures, 105
- make\_error\_condition
  - Diagnostics, 80
  - Futures, 105
- make\_exception\_ptr
  - Exceptions, 90
- make\_heap
  - Heap, 117, 118
- make\_index\_sequence
  - std, 688
- make\_integer\_sequence
  - std, 688
- make\_ostream\_joiner
  - experimental/iterator, 4133
- make\_pair
  - \_\_gnu\_parallel::\_IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1120
  - std::sub\_match< \_Biliter >, 3563
  - Utilities, 478
- make\_reverse\_iterator
  - Iterators, 139
- make\_shared
  - Pointer Abstractions, 311
- make\_signed\_t
  - Metaprogramming, 195
- make\_unique
  - Pointer Abstractions, 311, 312
- make\_unsigned\_t
  - Metaprogramming, 195
- malloc\_allocator.h, 4154
- map, 4154, 4155
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2905–2908
- map.h, 4156
- mapped\_type
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3726
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3765
- mark\_count
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1884
- mask\_array
  - Numeric Arrays, 265
- mask\_array.h, 4157
- mask\_based\_range\_hashing.hpp, 4157
- match\_any
  - std::regex\_constants, 851
- match\_continuous
  - std::regex\_constants, 851
- match\_default
  - std::regex\_constants, 852
- match\_flag\_type
  - std::regex\_constants, 841
- match\_not\_bol
  - std::regex\_constants, 852
- match\_not\_bow
  - std::regex\_constants, 852
- match\_not\_eol
  - std::regex\_constants, 852
- match\_not\_eow
  - std::regex\_constants, 852
- match\_not\_null
  - std::regex\_constants, 853
- match\_prev\_avail
  - std::regex\_constants, 853
- match\_results
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2941, 2942
- math.h, 4157
- Mathematical Special Functions, 147
  - airy\_ai, 151
  - airy\_aif, 151
  - airy\_ail, 152
  - airy\_bi, 152
  - airy\_bif, 152
  - airy\_bil, 152
  - assoc\_laguerre, 152
  - assoc\_laguerref, 154
  - assoc\_laguerrel, 154
  - assoc\_legendre, 154
  - assoc\_legendref, 155
  - assoc\_legendrel, 155
  - beta, 155

- betaf, 156
- betal, 156
- comp\_ellint\_1, 157
- comp\_ellint\_1f, 158
- comp\_ellint\_1l, 158
- comp\_ellint\_2, 158
- comp\_ellint\_2f, 159
- comp\_ellint\_2l, 159
- comp\_ellint\_3, 159
- comp\_ellint\_3f, 160
- comp\_ellint\_3l, 160
- conf\_hyperg, 161
- conf\_hypergf, 161
- conf\_hypergl, 162
- cyl\_bessel\_i, 162
- cyl\_bessel\_if, 163
- cyl\_bessel\_il, 163
- cyl\_bessel\_j, 163
- cyl\_bessel\_jf, 164
- cyl\_bessel\_jl, 164
- cyl\_bessel\_k, 164
- cyl\_bessel\_kf, 165
- cyl\_bessel\_kl, 165
- cyl\_neumann, 166
- cyl\_neumannf, 166
- cyl\_neumannl, 167
- ellint\_1, 167
- ellint\_1f, 168
- ellint\_1l, 168
- ellint\_2, 168
- ellint\_2f, 169
- ellint\_2l, 170
- ellint\_3, 170
- ellint\_3f, 171
- ellint\_3l, 171
- expint, 171
- expintf, 172
- expintl, 172
- hermite, 172
- hermitef, 173
- hermitel, 173
- hyperg, 174
- hypergf, 174
- hypergl, 175
- laguerre, 175
- laguerref, 176
- laguerrel, 176
- legendre, 176
- legendref, 177
- legendrel, 177
- riemann\_zeta, 178
- riemann\_zetaf, 178
- riemann\_zetal, 179
- sph\_bessel, 179
- sph\_besself, 180
- sph\_bessell, 180
- sph\_legendre, 180
- sph\_legendref, 181
- sph\_legendrel, 181
- sph\_neumann, 182
- sph\_neumannf, 183
- sph\_neumannl, 183
- max
  - \_\_gnu\_parallel, 571
  - Numeric Arrays, 272
  - Sorting, 419, 420
  - std::bernoulli\_distribution, 2100
  - std::binomial\_distribution< \_IntType >, 2139
  - std::cauchy\_distribution< \_RealType >, 2173
  - std::chi\_squared\_distribution< \_RealType >, 2201
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2448
  - std::discrete\_distribution< \_IntType >, 2453
  - std::exponential\_distribution< \_RealType >, 2526
  - std::extreme\_value\_distribution< \_RealType >, 2531
  - std::fisher\_f\_distribution< \_RealType >, 2540
  - std::gamma\_distribution< \_RealType >, 2606
  - std::geometric\_distribution< \_IntType >, 2611
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2688
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2820
  - std::lognormal\_distribution< \_RealType >, 2883
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2960
  - std::negative\_binomial\_distribution< \_IntType >, 3094
  - std::normal\_distribution< \_RealType >, 3103
  - std::numeric\_limits< \_Tp >, 3151
  - std::piecewise\_constant\_distribution< \_RealType >, 3260
  - std::piecewise\_linear\_distribution< \_RealType >, 3265
  - std::poisson\_distribution< \_IntType >, 3279
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3479
  - std::student\_t\_distribution< \_RealType >, 3553
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3569
  - std::uniform\_int\_distribution< \_IntType >, 3697
  - std::uniform\_real\_distribution< \_RealType >, 3701
  - std::weibull\_distribution< \_RealType >, 3930
- max\_bucket\_count
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3748
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3784

- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3821](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3855](#)
- max\_count
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, [2889](#)
- max\_digits10
  - std::\_\_numeric\_limits\_base, [952](#)
  - std::numeric\_limits< \_Tp >, [3155](#)
- max\_element
  - Sorting, [421](#)
- max\_element\_minimal\_n
  - \_\_gnu\_parallel:: Settings, [1281](#)
- max\_exponent
  - std::\_\_numeric\_limits\_base, [952](#)
  - std::numeric\_limits< \_Tp >, [3156](#)
- max\_exponent10
  - std::\_\_numeric\_limits\_base, [953](#)
  - std::numeric\_limits< \_Tp >, [3156](#)
- max\_load\_factor
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3748](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3785](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3821](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3856](#)
- max\_size
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [876](#)
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1022](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [1998](#)
  - std::allocator\_traits< \_Alloc >, [1312](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1319](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1955](#)
  - std::deque< \_Tp, \_Alloc >, [2430](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2567](#)
  - std::list< \_Tp, \_Alloc >, [2848](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2928](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2945](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3047](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3075](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3441](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2472](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3749](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3785](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3822](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3856](#)
  - std::vector< \_Tp, \_Alloc >, [3890](#)
- mean
  - std::normal\_distribution< \_RealType >, [3103](#)
  - std::poisson\_distribution< \_IntType >, [3279](#)
- mem\_fn
  - Function Objects, [101](#)
- Memory, [184](#)
  - align, [185](#)
  - uninitialized\_copy, [185](#)
  - uninitialized\_copy\_n, [186](#)
  - uninitialized\_fill, [186](#)
  - uninitialized\_fill\_n, [187](#)
- memory, [4158](#), [4159](#)
- memory\_order
  - Atomics, [25](#), [26](#)
- memory\_resource, [4160](#)
- memoryfwd.h, [4162](#)
- merge
  - Sorting, [422](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2567](#), [2568](#)
  - std::list< \_Tp, \_Alloc >, [2848](#), [2849](#)
- merge.h, [4162](#)
- merge\_minimal\_n
  - \_\_gnu\_parallel:: Settings, [1282](#)
- merge\_oversampling
  - \_\_gnu\_parallel:: Settings, [1282](#)
- mersenne\_twister\_engine
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2960](#)
- messages
  - std::locale, [2872](#)
  - std::messages< \_CharT >, [2965](#)
- messages\_members.h, [4163](#)
- metadata\_const\_reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [2106](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [2112](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1175](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_CIterator, Iterator, \_Alloc >, [1180](#)
- metadata\_reference
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, [2888](#)
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, [2893](#)
- metadata\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it<



- Node, Const\_Iterator, Iterator, \_Alloc >, [2106](#)
- \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it<  
Node, Const\_Iterator, Iterator, \_Alloc >, [2112](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer<  
Node, Leaf, Head, Inode, \_CIterator, Iterator,  
\_Alloc >, [1175](#)
- \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter<  
Node, Leaf, Head, Inode, \_CIterator, Iterator,  
\_Alloc >, [1180](#)
- \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc  
>, [2888](#)
- \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >,  
[2893](#)
- \_\_gnu\_pbds::sample\_update\_policy, [3402](#)
- Metaprogramming, [188](#)
- add\_const\_t, [192](#)
- add\_cv\_t, [192](#)
- add\_lvalue\_reference\_t, [193](#)
- add\_pointer\_t, [193](#)
- add\_rvalue\_reference\_t, [193](#)
- add\_volatile\_t, [193](#)
- aligned\_storage\_t, [193](#)
- alignment\_value, [197](#)
- common\_type\_t, [194](#)
- conditional\_t, [194](#)
- decay\_t, [194](#)
- enable\_if\_t, [194](#)
- false\_type, [194](#)
- make\_signed\_t, [195](#)
- make\_unsigned\_t, [195](#)
- remove\_all\_extents\_t, [195](#)
- remove\_const\_t, [195](#)
- remove\_cv\_t, [195](#)
- remove\_extent\_t, [196](#)
- remove\_pointer\_t, [196](#)
- remove\_reference\_t, [196](#)
- remove\_volatile\_t, [196](#)
- result\_of\_t, [196](#)
- true\_type, [197](#)
- underlying\_type\_t, [197](#)
- void\_t, [197](#)
- microseconds  
Time, [453](#)
- milliseconds  
Time, [453](#)
- min
  - \_\_gnu\_parallel, [571](#)
  - Numeric Arrays, [272](#)
  - Sorting, [424](#), [425](#)
  - std::bernoulli\_distribution, [2100](#)
  - std::binomial\_distribution< \_IntType >, [2139](#)
  - std::cauchy\_distribution< \_RealType >, [2174](#)
  - std::chi\_squared\_distribution< \_RealType >, [2201](#)
  - std::discard\_block\_engine< \_RandomNumberEngine,  
\_\_p, \_\_r >, [2448](#)
  - std::discrete\_distribution< \_IntType >, [2453](#)
  - std::exponential\_distribution< \_RealType >, [2526](#)
  - std::extreme\_value\_distribution< \_RealType >, [2531](#)
  - std::fisher\_f\_distribution< \_RealType >, [2541](#)
  - std::gamma\_distribution< \_RealType >, [2606](#)
  - std::geometric\_distribution< \_IntType >, [2611](#)
  - std::independent\_bits\_engine< \_RandomNum-  
berEngine, \_\_w, \_UIntType >, [2688](#)
  - std::linear\_congruential\_engine< \_UIntType, \_\_a,  
\_\_c, \_\_m >, [2820](#)
  - std::lognormal\_distribution< \_RealType >, [2884](#)
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w,  
\_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t,  
\_\_c, \_\_l, \_\_f >, [2960](#)
  - std::negative\_binomial\_distribution< \_IntType >,  
[3094](#)
  - std::normal\_distribution< \_RealType >, [3103](#)
  - std::numeric\_limits< \_Tp >, [3152](#)
  - std::piecewise\_constant\_distribution< \_RealType >,  
[3260](#)
  - std::piecewise\_linear\_distribution< \_RealType >,  
[3266](#)
  - std::poisson\_distribution< \_IntType >, [3279](#)
  - std::shuffle\_order\_engine< \_RandomNumberEngine,  
\_\_k >, [3479](#)
  - std::student\_t\_distribution< \_RealType >, [3553](#)
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w,  
\_\_s, \_\_r >, [3569](#)
  - std::uniform\_int\_distribution< \_IntType >, [3697](#)
  - std::uniform\_real\_distribution< \_RealType >, [3701](#)
  - std::weibull\_distribution< \_RealType >, [3930](#)
- min\_element  
Sorting, [425](#), [426](#)
- min\_element\_minimal\_n  
\_\_gnu\_parallel::\_Settings, [1282](#)
- min\_exponent  
std::\_\_numeric\_limits\_base, [953](#)  
std::numeric\_limits< \_Tp >, [3156](#)
- min\_exponent10  
std::\_\_numeric\_limits\_base, [953](#)  
std::numeric\_limits< \_Tp >, [3156](#)
- minmax  
Sorting, [426](#), [427](#)
- minmax\_element  
Sorting, [427](#), [428](#)
- minstd\_rand  
Random Number Generators, [336](#)
- minstd\_rand0  
Random Number Generators, [336](#)
- minutes  
Time, [453](#)
- mismatch



- Non-Mutating, [244–246](#)
- mod\_based\_range\_hashing.hpp, [4163](#)
- modulus
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2823](#)
- monetary
  - std::locale, [2872](#)
- money\_get
  - std::money\_get< \_CharT, \_InIter >, [2979](#)
- money\_put
  - std::money\_put< \_CharT, \_OutIter >, [2985](#)
- money\_punct
  - std::money\_punct< \_CharT, \_Intl >, [2992](#), [2993](#)
- move
  - Mutating, [205](#)
  - Utilities, [478](#)
- move.h, [4163](#)
- move\_backward
  - Mutating, [206](#)
- move\_if\_noexcept
  - Utilities, [479](#)
- mt19937
  - Random Number Generators, [336](#)
- mt19937\_64
  - Random Number Generators, [337](#)
- mt\_allocator.h, [4164](#)
- multimap
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3025–3028](#)
- multimap.h, [4165](#)
- multiplier
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2824](#)
- multiseq\_partition
  - \_\_gnu\_parallel, [572](#)
- multiseq\_selection
  - \_\_gnu\_parallel, [572](#)
- multiseq\_selection.h, [4166](#)
- multiset
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3057–3060](#)
- multiset.h, [4167](#)
- multiway\_merge
  - \_\_gnu\_parallel, [573](#)
- multiway\_merge.h, [4167](#)
  - \_GLIBCXX\_PARALLEL\_LENGTH, [4170](#)
- multiway\_merge\_3\_variant
  - \_\_gnu\_parallel, [575](#)
- multiway\_merge\_4\_variant
  - \_\_gnu\_parallel, [576](#)
- multiway\_merge\_exact\_splitting
  - \_\_gnu\_parallel, [576](#)
- multiway\_merge\_loser\_tree
  - \_\_gnu\_parallel, [577](#)
- multiway\_merge\_loser\_tree\_sentinel
  - \_\_gnu\_parallel, [578](#)
- multiway\_merge\_loser\_tree\_unguarded
  - \_\_gnu\_parallel, [578](#)
- multiway\_merge\_minimal\_k
  - \_\_gnu\_parallel::Settings, [1282](#)
- multiway\_merge\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1282](#)
- multiway\_merge\_oversampling
  - \_\_gnu\_parallel::Settings, [1283](#)
- multiway\_merge\_sampling\_splitting
  - \_\_gnu\_parallel, [579](#)
- multiway\_merge\_sentinels
  - \_\_gnu\_parallel, [580](#)
- multiway\_mergesort.h, [4170](#)
- Mutating, [198](#)
  - copy, [200](#)
  - copy\_backward, [200](#)
  - copy\_if, [201](#)
  - copy\_n, [201](#)
  - fill, [202](#)
  - fill\_n, [203](#)
  - generate, [203](#)
  - generate\_n, [204](#)
  - is\_partitioned, [204](#)
  - iter\_swap, [205](#)
  - move, [205](#)
  - move\_backward, [206](#)
  - partition, [207](#)
  - partition\_copy, [207](#)
  - partition\_point, [208](#)
  - random\_shuffle, [208](#)
  - remove, [209](#)
  - remove\_copy, [210](#)
  - remove\_copy\_if, [210](#)
  - remove\_if, [211](#)
  - replace, [211](#)
  - replace\_copy\_if, [212](#)
  - replace\_if, [213](#)
  - reverse, [213](#)
  - reverse\_copy, [214](#)
  - rotate, [214](#)
  - rotate\_copy, [215](#)
  - shuffle, [216](#)
  - stable\_partition, [216](#)
  - swap\_ranges, [217](#)
  - transform, [217](#), [218](#)
  - unique, [219](#)
  - unique\_copy, [220](#)
- mutex, [4171](#)
- Mutexes, [222](#)
  - adopt\_lock, [224](#)
  - call\_once, [223](#)
  - defer\_lock, [225](#)
  - lock, [223](#)

- try\_lock, [224](#)
- try\_to\_lock, [225](#)
- n
  - std::chi\_squared\_distribution< \_RealType >, [2201](#)
  - std::student\_t\_distribution< \_RealType >, [3553](#)
- name
  - std::locale, [2868](#)
  - std::type\_info, [3686](#)
- nanoseconds
  - Time, [454](#)
- narrow
  - std::\_\_ctype\_abstract\_base< \_CharT >, [909](#), [910](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1438](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1506](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1553](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1593](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1656](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1709](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1758](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1802](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1846](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2058](#)
  - std::ctype< \_CharT >, [2312](#)
  - std::ctype< char >, [2328](#), [2329](#)
  - std::ctype< wchar\_t >, [2348](#), [2349](#)
  - std::ctype\_byname< \_CharT >, [2365](#), [2366](#)
  - std::ctype\_byname< char >, [2381](#)
- native\_handle
  - std::thread, [3584](#)
- neg\_format
  - std::moneypunct< \_CharT, \_Intl >, [3000](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3011](#)
- negative\_sign
  - std::moneypunct< \_CharT, \_Intl >, [3000](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3012](#)
- Negators, [226](#)
  - not1, [226](#)
  - not2, [227](#)
- nested\_exception.h, [4172](#)
- net.h, [4172](#)
- Networking-ts, [228](#)
- new, [4172](#)
  - operator delete, [4173](#), [4174](#)
  - operator delete[], [4174](#), [4175](#)
  - operator new, [4175](#), [4176](#)
  - operator new[], [4176](#), [4177](#)
- new\_allocator.h, [4178](#)
- new\_handler
  - std, [688](#)
- next\_permutation
  - Sorting, [428](#), [429](#)
- noboolalpha
  - std, [723](#)
- node.hpp, [4178](#), [4179](#)
- node\_begin
  - \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3210](#), [3211](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [3254](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3331](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3490](#)
- node\_const\_iterator
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [2116](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, Node, \_Alloc >, [2117](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [3643](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [3645](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, Mapped, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3646](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, ov\_tree\_tag, \_Alloc >, [3647](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, rb\_tree\_tag, \_Alloc >, [3649](#)
  - \_\_gnu\_pbds::detail::tree\_traits< Key, null\_type, Cmp\_Fn, Node\_Update, splay\_tree\_tag, \_Alloc >, [3650](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3669](#)
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, [3670](#)
- node\_end
  - \_\_gnu\_pbds::detail::ov\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3211](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, [3254](#)
  - \_\_gnu\_pbds::detail::rb\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3331](#)
  - \_\_gnu\_pbds::detail::splay\_tree\_map< Key, Mapped, Cmp\_Fn, Node\_And\_It\_Traits, \_Alloc >, [3490](#)

- 3491
- node\_handle.h, 4179
- node\_iterators.hpp, 4179, 4180
- node\_metadata\_selector.hpp, 4180
- node\_type
  - \_\_gnu\_pbds::detail::pat\_trie\_base, 3251
  - \_\_gnu\_pbds::detail::pat\_trie\_map< Key, Mapped, Node\_And\_It\_Traits, \_Alloc >, 3254
- node\_update
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 3669
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 3670
- Non-Mutating, 229
  - adjacent\_find, 230, 231
  - all\_of, 231
  - any\_of, 232
  - count, 233
  - count\_if, 233
  - equal, 234, 235
  - find, 236
  - find\_end, 236, 237
  - find\_first\_of, 238
  - find\_if, 239
  - find\_if\_not, 240
  - for\_each, 240
  - is\_permutation, 241, 243
  - mismatch, 244–246
  - none\_of, 246
  - search, 247, 248
  - search\_n, 248, 249
- none
  - std::bitset< \_Nb >, 2158
  - std::locale, 2872
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2473
- none\_of
  - Non-Mutating, 246
- norm
  - Complex Numbers, 58
- Normal Distributions, 250
  - operator!=, 251, 252
  - operator<<, 253
  - operator>>, 253
- normal\_distribution
  - std::normal\_distribution< \_RealType >, 3102
- noshowbase
  - std, 724
- noshowpoint
  - std, 724
- noshowpos
  - std, 724
- noskipws
  - std, 724
- nosubs
  - std::regex\_constants, 853
- not1
  - Negators, 226
- not2
  - Negators, 227
- not\_fn
  - experimental/functional, 4102
- notify\_cleared
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2178
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2667
  - \_\_gnu\_pbds::sample\_resize\_policy, 3390
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3394
- notify\_erase\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2178
  - \_\_gnu\_pbds::sample\_resize\_policy, 3390
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3394
- notify\_erase\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2178
  - \_\_gnu\_pbds::sample\_resize\_policy, 3390
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3394
- notify\_erase\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2178
  - \_\_gnu\_pbds::sample\_resize\_policy, 3390
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3395
- notify\_erased
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2179
  - \_\_gnu\_pbds::sample\_resize\_policy, 3391
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3395
- notify\_externally\_resized
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2179
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3395
- notify\_find\_search\_collision
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2179
  - \_\_gnu\_pbds::sample\_resize\_policy, 3391
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3395
- notify\_find\_search\_end
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2179
  - \_\_gnu\_pbds::sample\_resize\_policy, 3391
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3395
- notify\_find\_search\_start
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, 2180
  - \_\_gnu\_pbds::sample\_resize\_policy, 3391

- `__gnu_pbds::sample_resize_trigger`, 3395
- `notify_insert_search_collision`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >`, 2180
  - `__gnu_pbds::sample_resize_policy`, 3391
  - `__gnu_pbds::sample_resize_trigger`, 3396
- `notify_insert_search_end`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger<std::num_put< _CharT, _OutIter >, 3135 External_Load_Access, Size_Type >`, 2180
  - `__gnu_pbds::sample_resize_policy`, 3391
  - `__gnu_pbds::sample_resize_trigger`, 3396
- `notify_insert_search_start`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger<~gslice, 269 External_Load_Access, Size_Type >`, 2180
  - `__gnu_pbds::sample_resize_policy`, 3392
  - `__gnu_pbds::sample_resize_trigger`, 3396
- `notify_inserted`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger<gslice, 264, 265 External_Load_Access, Size_Type >`, 2181
  - `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`, 2667
  - `__gnu_pbds::sample_resize_policy`, 3392
  - `__gnu_pbds::sample_resize_trigger`, 3396
- `notify_resized`
  - `__gnu_pbds::cc_hash_max_collision_check_resize_trigger<operator<=, 282, 283 External_Load_Access, Size_Type >`, 2181
  - `__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >`, 2668
  - `__gnu_pbds::sample_range_hashing`, 3385
  - `__gnu_pbds::sample_ranged_hash_fn`, 3387
  - `__gnu_pbds::sample_resize_policy`, 3392
  - `__gnu_pbds::sample_resize_trigger`, 3396
- `nounitbuf`
  - `std`, 725
- `nouppercase`
  - `std`, 725
- `npos`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1044
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 2006
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1975
- `nth_element`
  - Sorting, 429, 430
- `nth_element_minimal_n`
  - `__gnu_parallel::Settings`, 1283
- `null_node_metadata.hpp`, 4181
- `nullopt`
  - Optional values, 303
- `num_blocks`
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2473
- `num_children`
  - `__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1176
- `__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _Clterator, Iterator, _Alloc >`, 1181
- `num_get`
  - `std::num_get< _CharT, _InIter >`, 3114
- `num_put`
- `numbers`, 4181
- `numeric`, 4181–4183, 4185
  - `std::locale`, 2873
- `Numeric Arrays`, 255
  - `apply`, 269, 270
  - `begin`, 270
  - `cshift`, 271
  - `end`, 271, 272
  - `gslice_array`, 265
  - `indirect_array`, 265
  - `mask_array`, 265
  - `max`, 272
  - `min`, 272
  - `operator!`, 272
  - `operator<=`, 282, 283
  - `operator>=`, 290, 291
  - `operator*=`, 276, 277
  - `operator~`, 299
  - `operator^=`, 296, 297
  - `operator+`, 277
  - `operator+=`, 277–279
  - `operator-`, 279
  - `operator-=`, 279, 280
  - `operator/=`, 281, 282
  - `operator=`, 284–289
  - `operator%=`, 273, 274
  - `operator&=`, 274, 275
  - `operator[]`, 291–295
  - `operator|=`, 297, 298
  - `resize`, 299
  - `shift`, 299
  - `size`, 300
  - `slice`, 266
  - `slice_array`, 266
  - `start`, 300, 301
  - `stride`, 301
  - `sum`, 301
  - `swap`, 301
  - `valarray`, 267–269
- `numeric_traits.h`, 4186
- `numericfwd.h`, 4186
- `Numerics`, 302
- `numpunct`
  - `std::numpunct< _CharT >`, 3178, 3179

## oct

std, 725  
 std::basic\_fstream< \_CharT, \_Traits >, 1477  
 std::basic\_ifstream< \_CharT, \_Traits >, 1533  
 std::basic\_ios< \_CharT, \_Traits >, 1566  
 std::basic\_iostream< \_CharT, \_Traits >, 1631  
 std::basic\_istream< \_CharT, \_Traits >, 1683  
 std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1737  
 std::basic\_ofstream< \_CharT, \_Traits >, 1782  
 std::basic\_ostream< \_CharT, \_Traits >, 1825  
 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1870  
 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2095  
 std::ios\_base, 2721

## off\_type

std::basic\_ios< \_CharT, \_Traits >, 1543  
 std::basic\_streambuf< \_CharT, \_Traits >, 1891  
 std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3909

## ofstream

I/O, 128

## omp\_loop.h, 4188

## omp\_loop\_static.h, 4188

## once\_flag

std::once\_flag, 3193

## open

\_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2490, 2491  
 \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3505, 3506  
 std::basic\_filebuf< \_CharT, \_Traits >, 1393, 1394  
 std::basic\_fstream< \_CharT, \_Traits >, 1439  
 std::basic\_ifstream< \_CharT, \_Traits >, 1506, 1507  
 std::basic\_ofstream< \_CharT, \_Traits >, 1758, 1759

## openmode

std::basic\_fstream< \_CharT, \_Traits >, 1421  
 std::basic\_ifstream< \_CharT, \_Traits >, 1489  
 std::basic\_ios< \_CharT, \_Traits >, 1543  
 std::basic\_iostream< \_CharT, \_Traits >, 1577  
 std::basic\_istream< \_CharT, \_Traits >, 1641  
 std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1693  
 std::basic\_ofstream< \_CharT, \_Traits >, 1747  
 std::basic\_ostream< \_CharT, \_Traits >, 1792  
 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1835  
 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2041  
 std::ios\_base, 2708

## operator\_iterator

\_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 1226  
 \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1244

## operator\_RAlter

\_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 1086

## operator bool

std::basic\_fstream< \_CharT, \_Traits >, 1440  
 std::basic\_ifstream< \_CharT, \_Traits >, 1507  
 std::basic\_ios< \_CharT, \_Traits >, 1553  
 std::basic\_iostream< \_CharT, \_Traits >, 1594  
 std::basic\_istream< \_CharT, \_Traits >, 1657  
 std::basic\_istream< \_CharT, \_Traits >::sentry, 3413  
 std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1710  
 std::basic\_ofstream< \_CharT, \_Traits >, 1759  
 std::basic\_ostream< \_CharT, \_Traits >, 1802  
 std::basic\_ostream< \_CharT, \_Traits >::sentry, 3415  
 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1846  
 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2058  
 std::function< \_Res(\_ArgTypes...)>, 2589  
 std::shared\_ptr< \_Tp >, 3473  
 std::tr2::bool\_set, 2169  
 std::unique\_ptr< \_Tp, \_Dp >, 3710  
 std::unique\_ptr< \_Tp[], \_Dp >, 3717

## operator delete

new, 4173, 4174

## operator delete[]

new, 4174, 4175

## operator new

new, 4175, 4176

## operator new[]

new, 4176, 4177

## operator streamoff

std::fpos< \_StateT >, 2577

## operator string\_type

std::sub\_match< \_Biter >, 3562

## operator!

Numeric Arrays, 272

std::basic\_fstream< \_CharT, \_Traits >, 1440  
 std::basic\_ifstream< \_CharT, \_Traits >, 1507  
 std::basic\_ios< \_CharT, \_Traits >, 1553  
 std::basic\_iostream< \_CharT, \_Traits >, 1594  
 std::basic\_istream< \_CharT, \_Traits >, 1657  
 std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1710  
 std::basic\_ofstream< \_CharT, \_Traits >, 1759  
 std::basic\_ostream< \_CharT, \_Traits >, 1802  
 std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1846  
 std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2058  
 operator!=  
 \_\_gnu\_cxx, 500, 501

- `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 1121
- `__gnu_pbds::detail::bin_search_tree_const_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2108
- `__gnu_pbds::detail::bin_search_tree_node_it< Node, Const_Iterator, Iterator, _Alloc >`, 2114
- `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 2126
- `__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 2131
- `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, 2803
- `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, 2809
- `__gnu_pbds::detail::pat_trie_base::Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`, 1176
- `__gnu_pbds::detail::pat_trie_base::Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >`, 1181
- Allocators, 8
- Bernoulli Distributions, 36
- Complex Numbers, 59
- Diagnostics, 80
- Dynamic Bitset., 83
- Normal Distributions, 251, 252
- Pointer Abstractions, 312, 313
- Poisson Distributions, 326, 327
- Random Number Generators, 337–339
- Regular Expressions, 351, 352, 354, 355
- std, 725–730
- `std::_Fwd_list_const_iterator< _Tp >`, 1080
- `std::_Fwd_list_iterator< _Tp >`, 1082
- `std::bitset< _Nb >`, 2158
- `std::istream_iterator< _Tp, _CharT, _Traits, _Dist >`, 2785
- `std::locale`, 2868
- `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3347
- `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3353
- `std::rel_ops`, 854
- `std::sub_match< _Bilter >`, 3564
- Uniform Distributions, 464, 465
- Utilities, 480
- `operator<`
  - `__gnu_cxx`, 504, 505
  - `__gnu_parallel::GuardedIterator< _RAIter, _Compare >`, 1087
  - `__gnu_parallel::IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >`, 1121
  - Diagnostics, 81
  - Pointer Abstractions, 314, 315
  - Regular Expressions, 355–358
  - std, 733–740
  - `std::sub_match< _Bilter >`, 3564
  - Utilities, 480
- `operator<<`
  - Bernoulli Distributions, 36, 38
  - Complex Numbers, 64
  - Dynamic Bitset., 84
  - Normal Distributions, 253
  - Pointer Abstractions, 315
  - Poisson Distributions, 328, 329
  - Random Number Generators, 340
  - Regular Expressions, 358
  - `std::__detail`, 806
  - `std::basic_fstream< _CharT, _Traits >`, 1440–1448
  - `std::basic_iostream< _CharT, _Traits >`, 1594–1596, 1598–1602
  - `std::basic_ofstream< _CharT, _Traits >`, 1759–1767
  - `std::basic_ostream< _CharT, _Traits >`, 1803–1810
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1847–1854
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2059–2066
  - `std::binomial_distribution< _IntType >`, 2142
  - `std::bitset< _Nb >`, 2159
  - `std::chi_squared_distribution< _RealType >`, 2203
  - `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2450
  - `std::discrete_distribution< _IntType >`, 2455
  - `std::fisher_f_distribution< _RealType >`, 2542
  - `std::gamma_distribution< _RealType >`, 2608
  - `std::linear_congruential_engine< _UIntType, __a, __c, __m >`, 2822
  - `std::lognormal_distribution< _RealType >`, 2885
  - `std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >`, 2961
  - `std::negative_binomial_distribution< _IntType >`, 3096
  - `std::normal_distribution< _RealType >`, 3105
  - `std::piecewise_constant_distribution< _RealType >`, 3262
  - `std::piecewise_linear_distribution< _RealType >`, 3267
  - `std::poisson_distribution< _IntType >`, 3281
  - `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 3481
  - `std::student_t_distribution< _RealType >`, 3555
  - `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 3570
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2474
  - Uniform Distributions, 465
- `operator<=>`
  - Numeric Arrays, 282, 283



- std::bitset< \_Nb >, 2159
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2474
- operator<=
  - \_\_gnu\_cxx, 505, 506
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 1087
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1121
  - Dynamic Bitset., 84
  - Pointer Abstractions, 315–317
  - Regular Expressions, 359, 361–363
  - std, 748–751
  - std::\_\_debug, 800
  - std::rel\_ops, 854
  - std::sub\_match< \_Biliter >, 3564
  - Utilities, 480
- operator>
  - \_\_gnu\_cxx, 509, 510
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1122
  - Dynamic Bitset., 84
  - Pointer Abstractions, 318–320
  - Regular Expressions, 367, 368, 370, 371
  - std, 761–764
  - std::\_\_debug, 800
  - std::rel\_ops, 855
  - std::sub\_match< \_Biliter >, 3565
  - Utilities, 481
- operator>>
  - Bernoulli Distributions, 38, 39
  - Complex Numbers, 66
  - Dynamic Bitset., 85
  - Normal Distributions, 253
  - Poisson Distributions, 329, 330
  - std, 768, 770–776
  - std::\_\_detail, 806
  - std::basic\_fstream< \_CharT, \_Traits >, 1448–1456
  - std::basic\_ifstream< \_CharT, \_Traits >, 1508–1515
  - std::basic\_iostream< \_CharT, \_Traits >, 1603–1610
  - std::basic\_istream< \_CharT, \_Traits >, 1657–1659, 1661–1665
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1710–1712, 1714–1718
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2066–2074
  - std::binomial\_distribution< \_IntType >, 2142
  - std::bitset< \_Nb >, 2160
  - std::chi\_squared\_distribution< \_RealType >, 2203
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2451
  - std::discrete\_distribution< \_IntType >, 2455
  - std::fisher\_f\_distribution< \_RealType >, 2543
  - std::gamma\_distribution< \_RealType >, 2609
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2690
  - std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2823
  - std::lognormal\_distribution< \_RealType >, 2886
  - std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2962
  - std::negative\_binomial\_distribution< \_IntType >, 3096
  - std::normal\_distribution< \_RealType >, 3106
  - std::piecewise\_constant\_distribution< \_RealType >, 3262
  - std::piecewise\_linear\_distribution< \_RealType >, 3268
  - std::poisson\_distribution< \_IntType >, 3282
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3482
  - std::student\_t\_distribution< \_RealType >, 3555
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3571
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2475
  - Uniform Distributions, 466
- operator>>=
  - Numeric Arrays, 290, 291
  - std::bitset< \_Nb >, 2160
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2475
- operator\*=
  - \_\_gnu\_cxx, 510, 511
  - \_\_gnu\_parallel::IteratorPair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1122
  - Dynamic Bitset., 85
  - Pointer Abstractions, 320, 321
  - Regular Expressions, 372, 373, 375
  - std, 765–768
  - std::\_\_debug, 801
  - std::rel\_ops, 855
  - std::sub\_match< \_Biliter >, 3565
  - Utilities, 481
- operator\*
  - \_\_gnu\_debug::Safe\_iterator< \_Iterator, \_Sequence, \_Category >, 1226
  - \_\_gnu\_debug::Safe\_local\_iterator< \_Iterator, \_Sequence >, 1244
  - \_\_gnu\_parallel::GuardedIterator< \_RAIter, \_Compare >, 1086
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 2108
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 2114
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2126
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2131

- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2803
- \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 2809
- \_\_gnu\_pbds::detail::ov\_tree\_node\_it< Value\_Type, Metadata\_Type, \_Alloc >, 3215
- \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1176
- \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_Clterator, Iterator, \_Alloc >, 1181
- Complex Numbers, 59, 60
- std::auto\_ptr< \_Tp >, 1364
- std::back\_insert\_iterator< \_Container >, 1370
- std::front\_insert\_iterator< \_Container >, 2584
- std::insert\_iterator< \_Container >, 2700
- std::istreambuf\_iterator< \_CharT, \_Traits >, 2790
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3205
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3347
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3353
- std::reverse\_iterator< \_Iterator >, 3372
- std::unique\_ptr< \_Tp, \_Dp >, 3710
- Time, 454, 455
- operator\*=
  - Complex Numbers, 60
  - Numeric Arrays, 276, 277
- operator~
  - Numeric Arrays, 299
  - std::bitset< \_Nb >, 2162
  - std::regex\_constants, 847, 848
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2477
- operator^
  - Dynamic Bitset., 85
  - std, 777
  - std::regex\_constants, 845
- operator^=
  - Numeric Arrays, 296, 297
  - std::bitset< \_Nb >, 2162
  - std::regex\_constants, 845, 846
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2477
- operator()
  - \_\_gnu\_cxx::subtractive\_rng, 3573
  - \_\_gnu\_parallel::Nothing, 1185
  - \_\_gnu\_parallel::RandomNumber, 1202, 1203
  - \_\_gnu\_parallel::accumulate\_selector< \_It >, 863
  - \_\_gnu\_parallel::adjacent\_find\_selector, 867
  - \_\_gnu\_parallel::count\_if\_selector< \_It, \_Diff >, 896
  - \_\_gnu\_parallel::count\_selector< \_It, \_Diff >, 897
  - \_\_gnu\_parallel::fill\_selector< \_It >, 920
  - \_\_gnu\_parallel::find\_first\_of\_selector< \_FIterator, \_Tp, \_Diff >, 924
  - \_\_gnu\_parallel::find\_if\_selector, 924
  - \_\_gnu\_parallel::for\_each\_selector< \_It >, 925
  - \_\_gnu\_parallel::generate\_selector< \_It >, 929
  - \_\_gnu\_parallel::identity\_selector< \_It >, 933
  - \_\_gnu\_parallel::inner\_product\_selector< \_It, \_It2, \_Tp >, 935
  - \_\_gnu\_parallel::mismatch\_selector, 942
  - \_\_gnu\_parallel::replace\_if\_selector< \_It, \_Op, \_Tp >, 966
  - \_\_gnu\_parallel::replace\_selector< \_It, \_Tp >, 968
  - \_\_gnu\_parallel::transform1\_selector< \_It >, 971
  - \_\_gnu\_parallel::transform2\_selector< \_It >, 973
  - \_\_gnu\_pbds::direct\_mask\_range\_hashing< Size\_Type >, 2443
  - \_\_gnu\_pbds::direct\_mod\_range\_hashing< Size\_Type >, 2444
  - \_\_gnu\_pbds::linear\_probe\_fn< Size\_Type >, 2825
  - \_\_gnu\_pbds::lu\_counter\_policy< Max\_Count, \_Alloc >, 2889
  - \_\_gnu\_pbds::lu\_move\_to\_front\_policy< \_Alloc >, 2893
  - \_\_gnu\_pbds::quadratic\_probe\_fn< Size\_Type >, 3296
  - \_\_gnu\_pbds::sample\_probe\_fn, 3383
  - \_\_gnu\_pbds::sample\_range\_hashing, 3385
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 3387
  - \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_ltr, \_ATraits, \_Alloc >, 3401
  - \_\_gnu\_pbds::sample\_update\_policy, 3403
  - \_\_gnu\_pbds::tree\_order\_statistics\_node\_update< Node\_Cltr, Node\_ltr, Cmp\_Fn, \_Alloc >, 3641
  - \_\_gnu\_pbds::trie\_order\_statistics\_node\_update< Node\_Cltr, Node\_ltr, \_ATraits, \_Alloc >, 3657
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_ltr, \_ATraits, \_Alloc >, 3663
  - std::bernoulli\_distribution, 2100
  - std::binomial\_distribution< \_IntType >, 2140
  - std::cauchy\_distribution< \_RealType >, 2174
  - std::chi\_squared\_distribution< \_RealType >, 2201
  - std::default\_delete< \_Tp >, 2397
  - std::default\_delete< \_Tp[] >, 2398
  - std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2449
  - std::discrete\_distribution< \_IntType >, 2453
  - std::exponential\_distribution< \_RealType >, 2527
  - std::extreme\_value\_distribution< \_RealType >, 2532
  - std::fisher\_f\_distribution< \_RealType >, 2541
  - std::function< \_Res(\_ArgTypes...) >, 2589
  - std::gamma\_distribution< \_RealType >, 2607
  - std::geometric\_distribution< \_IntType >, 2611
  - std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2689
  - std::linear\_congruential\_engine< \_UIntType, \_\_a,



- `__c, __m >`, 2820
- `std::locale`, 2868
- `std::lognormal_distribution< _RealType >`, 2884
- `std::negative_binomial_distribution< _IntType >`, 3094
- `std::normal_distribution< _RealType >`, 3103, 3104
- `std::piecewise_constant_distribution< _RealType >`, 3260
- `std::piecewise_linear_distribution< _RealType >`, 3266
- `std::poisson_distribution< _IntType >`, 3279, 3280
- `std::shuffle_order_engine< _RandomNumberEngine, __k >`, 3480
- `std::student_t_distribution< _RealType >`, 3553
- `std::subtract_with_carry_engine< _UIntType, __w, __s, __r >`, 3569
- `std::uniform_int_distribution< _IntType >`, 3697
- `std::uniform_real_distribution< _RealType >`, 3701
- `std::weibull_distribution< _RealType >`, 3931
- operator+
  - `__gnu_cxx`, 501–503
  - Complex Numbers, 61
  - Numeric Arrays, 277
  - `std`, 731–733
  - `std::fpos< _StateT >`, 2577
  - `std::reverse_iterator< _Iterator >`, 3373
  - Time, 455
- operator++
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1227
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1244, 1245
  - `__gnu_parallel::GuardedIterator< _RAIter, _Compare >`, 1086
  - `std::back_insert_iterator< _Container >`, 1371
  - `std::front_insert_iterator< _Container >`, 2584
  - `std::insert_iterator< _Container >`, 2700
  - `std::istreambuf_iterator< _CharT, _Traits >`, 2790, 2791
  - `std::ostreambuf_iterator< _CharT, _Traits >`, 3205, 3206
  - `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3348
  - `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3353
  - `std::reverse_iterator< _Iterator >`, 3373
- operator+=
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1022, 1024, 1025
  - `__gnu_debug::basic_string< _CharT, _Traits, _Allocator >`, 1998
  - Complex Numbers, 62
  - Numeric Arrays, 277–279
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1955–1957
  - `std::complex< _Tp >`, 2259
  - `std::fpos< _StateT >`, 2577
  - `std::reverse_iterator< _Iterator >`, 3374
- operator-
  - Complex Numbers, 62, 63
  - Dynamic Bitset., 84
  - Numeric Arrays, 279
  - `std::fpos< _StateT >`, 2578
  - `std::reverse_iterator< _Iterator >`, 3374
  - Time, 456
- operator->
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1227
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1245
  - `__gnu_pbds::detail::binary_heap_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 2126
  - `__gnu_pbds::detail::binary_heap_point_const_iterator< Value_Type, Entry, Simple, _Alloc >`, 2131
  - `__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator< Node, _Alloc >`, 2804
  - `__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >`, 2809
  - `std::auto_ptr< _Tp >`, 1365
  - `std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3348
  - `std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >`, 3354
  - `std::reverse_iterator< _Iterator >`, 3375
  - `std::unique_ptr< _Tp, _Dp >`, 3710
- operator--
  - `std::reverse_iterator< _Iterator >`, 3374, 3375
- operator-=
  - Complex Numbers, 63
  - Numeric Arrays, 279, 280
  - `std::complex< _Tp >`, 2259
  - `std::fpos< _StateT >`, 2578
  - `std::reverse_iterator< _Iterator >`, 3375
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2474
- operator/
  - Complex Numbers, 63, 64
- operator/=
  - Complex Numbers, 64
  - Numeric Arrays, 281, 282
- operator=
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1025–1027
  - `__gnu_debug::Safe_iterator< _Iterator, _Sequence, _Category >`, 1228
  - `__gnu_debug::Safe_local_iterator< _Iterator, _Sequence >`, 1245, 1246
  - Complex Numbers, 65
  - Numeric Arrays, 284–289

std::\_\_allocated\_ptr< \_Alloc >, 879  
 std::auto\_ptr< \_Tp >, 1365  
 std::back\_insert\_iterator< \_Container >, 1371  
 std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1884, 1885  
 std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1957–1959  
 std::deque< \_Tp, \_Alloc >, 2430, 2431  
 std::experimental::fundamentals\_v1::any, 1324, 1325  
 std::forward\_list< \_Tp, \_Alloc >, 2568, 2569  
 std::front\_insert\_iterator< \_Container >, 2584  
 std::function< \_Res(\_ArgTypes...)>, 2590–2592  
 std::insert\_iterator< \_Container >, 2700  
 std::list< \_Tp, \_Alloc >, 2849, 2850  
 std::locale, 2869  
 std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2928, 2929  
 std::match\_results< \_Bi\_iter, \_Alloc >, 2945, 2946  
 std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3047  
 std::multiset< \_Key, \_Compare, \_Alloc >, 3075, 3076  
 std::once\_flag, 3194  
 std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3200  
 std::ostreambuf\_iterator< \_CharT, \_Traits >, 3206  
 std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3348  
 std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3354  
 std::set< \_Key, \_Compare, \_Alloc >, 3441, 3442  
 std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2475  
 std::unique\_ptr< \_Tp, \_Dp >, 3711  
 std::unique\_ptr< \_Tp[], \_Dp >, 3717, 3718  
 std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3749  
 std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3786  
 std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3822  
 std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3857  
 std::vector< \_Tp, \_Alloc >, 3890, 3891  
 operator==  
   \_\_gnu\_cxx, 507, 508  
   \_\_gnu\_parallel::iterator\_pair< \_Iterator1, \_Iterator2, \_IteratorCategory >, 1122  
   \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, 2108  
   \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_iterator< Node, Const\_Iterator, Iterator, \_Alloc >, 2114  
   \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2127  
   \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2132  
   \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2804  
   \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 2810  
   \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_citer< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, 1176  
   \_\_gnu\_pbds::detail::pat\_trie\_base::Node\_iter< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, 1181  
 Allocators, 8  
 Complex Numbers, 65, 66  
 Diagnostics, 81  
 Iterators, 139  
 Pointer Abstractions, 317, 318  
 Regular Expressions, 363, 364, 366, 367  
 std, 752–755, 757–760  
 std::\_\_exception\_ptr::exception\_ptr, 2524  
 std::\_Fwd\_list\_const\_iterator< \_Tp >, 1081  
 std::\_Fwd\_list\_iterator< \_Tp >, 1082  
 std::bernoulli\_distribution, 2101  
 std::binomial\_distribution< \_IntType >, 2142  
 std::bitset< \_Nb >, 2160  
 std::cauchy\_distribution< \_RealType >, 2175  
 std::chi\_squared\_distribution< \_RealType >, 2203  
 std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, 2450  
 std::discrete\_distribution< \_IntType >, 2455  
 std::exponential\_distribution< \_RealType >, 2528  
 std::extreme\_value\_distribution< \_RealType >, 2533  
 std::fisher\_f\_distribution< \_RealType >, 2542  
 std::gamma\_distribution< \_RealType >, 2608  
 std::geometric\_distribution< \_IntType >, 2613  
 std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2690  
 std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2785  
 std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2822  
 std::locale, 2869  
 std::lognormal\_distribution< \_RealType >, 2885  
 std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, 2961  
 std::negative\_binomial\_distribution< \_IntType >, 3096  
 std::normal\_distribution< \_RealType >, 3106  
 std::piecewise\_constant\_distribution< \_RealType >, 3262  
 std::piecewise\_linear\_distribution< \_RealType >, 3267  
 std::poisson\_distribution< \_IntType >, 3281  
 std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3349  
 std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type,

- `_Rx_traits` >, 3354
  - `std::shuffle_order_engine` < `_RandomNumberEngine`, `__k` >, 3481
  - `std::student_t_distribution` < `_RealType` >, 3555
  - `std::sub_match` < `_Bilter` >, 3564
  - `std::subtract_with_carry_engine` < `_UIntType`, `__w`, `__s`, `__r` >, 3571
  - `std::uniform_int_distribution` < `_IntType` >, 3699
  - `std::uniform_real_distribution` < `_RealType` >, 3702
  - `std::weibull_distribution` < `_RealType` >, 3932
  - Utilities, 480
- `operator%=>`
  - Numeric Arrays, 273, 274
- `operator&`
  - Dynamic Bitset., 83
  - `std`, 730
  - `std::regex_constants`, 843, 844
- `operator&=`
  - Numeric Arrays, 274, 275
  - `std::bitset` < `_Nb` >, 2159
  - `std::regex_constants`, 844
  - `std::tr2::dynamic_bitset` < `_WordT`, `_Alloc` >, 2473, 2474
- `operator""h`
  - `std::literals::chrono_literals`, 835
- `operator""min`
  - `std::literals::chrono_literals`, 835, 836
- `operator""ms`
  - `std::literals::chrono_literals`, 836
- `operator""ns`
  - `std::literals::chrono_literals`, 836
- `operator""s`
  - `std::literals::chrono_literals`, 837
- `operator""us`
  - `std::literals::chrono_literals`, 837
- `operator[]`
  - `__gnu_cxx::__versa_string` < `_CharT`, `_Traits`, `_Alloc`, `_Base` >, 1027, 1028
  - Numeric Arrays, 291–295
  - `std::basic_string` < `_CharT`, `_Traits`, `_Alloc` >, 1959
  - `std::bitset` < `_Nb` >, 2161
  - `std::deque` < `_Tp`, `_Alloc` >, 2431, 2432
  - `std::map` < `_Key`, `_Tp`, `_Compare`, `_Alloc` >, 2929
  - `std::match_results` < `_Bi_iter`, `_Alloc` >, 2946
  - `std::reverse_iterator` < `_Iterator` >, 3376
  - `std::tr2::dynamic_bitset` < `_WordT`, `_Alloc` >, 2476
  - `std::unique_ptr` < `_Tp[]`, `_Dp` >, 3718
  - `std::unordered_map` < `_Key`, `_Tp`, `_Hash`, `_Pred`, `_Alloc` >, 3750
  - `std::vector` < `_Tp`, `_Alloc` >, 3891, 3892
- `operator|`
  - Dynamic Bitset., 86
  - `std`, 777
  - `std::regex_constants`, 846
- `operator|=`
  - Numeric Arrays, 297, 298
  - `std::bitset` < `_Nb` >, 2162
  - `std::regex_constants`, 847
  - `std::tr2::dynamic_bitset` < `_WordT`, `_Alloc` >, 2477
- `opt_random.h`, 4189
- `optimize`
  - `std::regex_constants`, 853
- optional, 4189
- Optional values, 303
  - `in_place`, 303
  - `nullopt`, 303
- `order_of_key`
  - `__gnu_pbds::tree_order_statistics_node_update` < `Node_Cltr`, `Node_Itr`, `Cmp_Fn`, `_Alloc` >, 3641
  - `__gnu_pbds::trie_order_statistics_node_update` < `Node_Cltr`, `Node_Itr`, `_ATraits`, `_Alloc` >, 3657
- `order_of_prefix`
  - `__gnu_pbds::trie_order_statistics_node_update` < `Node_Cltr`, `Node_Itr`, `_ATraits`, `_Alloc` >, 3658
- `order_preserving`
  - `__gnu_pbds::container_traits` < `Cntr` >, 2294
- `order_statistics_imp.hpp`, 4190
- `os_defines.h`, 4190
- `ostream`, 4191
  - I/O, 128
- `ostream.tcc`, 4192
- `ostream_insert.h`, 4193
- `ostream_iterator`
  - `std::ostream_iterator` < `_Tp`, `_CharT`, `_Traits` >, 3199, 3200
- `ostream_type`
  - `std::ostream_iterator` < `_Tp`, `_CharT`, `_Traits` >, 3198
  - `std::ostreambuf_iterator` < `_CharT`, `_Traits` >, 3203
- `ostreambuf_iterator`
  - `std::ostreambuf_iterator` < `_CharT`, `_Traits` >, 3204, 3205
- `ostreamstring`
  - I/O, 128
- `out`
  - `std::__codecvt_abstract_base` < `_InternT`, `_ExternT`, `_StateT` >, 892
  - `std::basic_fstream` < `_CharT`, `_Traits` >, 1477
  - `std::basic_ifstream` < `_CharT`, `_Traits` >, 1533
  - `std::basic_ios` < `_CharT`, `_Traits` >, 1566
  - `std::basic_iostream` < `_CharT`, `_Traits` >, 1631
  - `std::basic_istream` < `_CharT`, `_Traits` >, 1683
  - `std::basic_istreamstream` < `_CharT`, `_Traits`, `_Alloc` >, 1737
  - `std::basic_ofstream` < `_CharT`, `_Traits` >, 1782
  - `std::basic_ostream` < `_CharT`, `_Traits` >, 1826
  - `std::basic_ostringstream` < `_CharT`, `_Traits`, `_Alloc` >, 1870

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2095
- std::codecvt< \_InternT, \_ExternT, \_StateT >, 2207
- std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2213
- std::codecvt< char, char, mbstate\_t >, 2218
- std::codecvt< char16\_t, char, mbstate\_t >, 2223
- std::codecvt< char32\_t, char, mbstate\_t >, 2228
- std::codecvt< wchar\_t, char, mbstate\_t >, 2233
- std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2239
- std::ios\_base, 2722
- ov\_tree\_map.hpp, 4193
- overflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2492
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3507
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3530
  - std::basic\_filebuf< \_CharT, \_Traits >, 1395
  - std::basic\_streambuf< \_CharT, \_Traits >, 1896
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2016
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3913
- owner\_before
  - std::shared\_ptr< \_Tp >, 3473, 3474
- p
  - std::bernoulli\_distribution, 2100
  - std::binomial\_distribution< \_IntType >, 2140
  - std::geometric\_distribution< \_IntType >, 2611
  - std::negative\_binomial\_distribution< \_IntType >, 3095
- pair
  - std::pair< \_T1, \_T2 >, 3228, 3229
  - Utilities, 472
- pairing\_heap.hpp, 4194
- par\_loop.h, 4194
- parallel.h, 4195
- parallel\_balanced
  - \_\_gnu\_parallel, 534
- parallel\_multiway\_merge
  - \_\_gnu\_parallel, 582
- parallel\_omp\_loop
  - \_\_gnu\_parallel, 534
- parallel\_omp\_loop\_static
  - \_\_gnu\_parallel, 534
- parallel\_sort\_mwms
  - \_\_gnu\_parallel, 582
- parallel\_sort\_mwms\_pu
  - \_\_gnu\_parallel, 583
- parallel\_tag
  - \_\_gnu\_parallel::parallel\_tag, 3235
- parallel\_taskqueue
  - \_\_gnu\_parallel, 534
- parallel\_unbalanced
  - \_\_gnu\_parallel, 534
- param
  - std::bernoulli\_distribution, 2101
  - std::binomial\_distribution< \_IntType >, 2140, 2141
  - std::cauchy\_distribution< \_RealType >, 2174
  - std::chi\_squared\_distribution< \_RealType >, 2202
  - std::discrete\_distribution< \_IntType >, 2453, 2454
  - std::exponential\_distribution< \_RealType >, 2527
  - std::extreme\_value\_distribution< \_RealType >, 2532
  - std::fisher\_f\_distribution< \_RealType >, 2541
  - std::gamma\_distribution< \_RealType >, 2607
  - std::geometric\_distribution< \_IntType >, 2612
  - std::lognormal\_distribution< \_RealType >, 2884
  - std::negative\_binomial\_distribution< \_IntType >, 3095
  - std::normal\_distribution< \_RealType >, 3104
  - std::piecewise\_constant\_distribution< \_RealType >, 3261
  - std::piecewise\_linear\_distribution< \_RealType >, 3266
  - std::poisson\_distribution< \_IntType >, 3280
  - std::student\_t\_distribution< \_RealType >, 3554
  - std::uniform\_int\_distribution< \_IntType >, 3698
  - std::uniform\_real\_distribution< \_RealType >, 3701, 3702
  - std::weibull\_distribution< \_RealType >, 3931
- parse\_numbers.h, 4195
- partial\_sort
  - Sorting, 430, 431
- partial\_sort\_copy
  - Sorting, 432
- partial\_sort\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1283
- partial\_sum
  - Generalized Numeric operations, 111
- partial\_sum.h, 4195
- partial\_sum\_dilation
  - \_\_gnu\_parallel::Settings, 1283
- partial\_sum\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1283
- partition
  - Mutating, 207
- partition.h, 4196
  - \_GLIBCXX\_VOLATILE, 4196
- partition\_chunk\_share
  - \_\_gnu\_parallel::Settings, 1284
- partition\_chunk\_size
  - \_\_gnu\_parallel::Settings, 1284
- partition\_copy
  - Mutating, 207
- partition\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1284
- partition\_point
  - Mutating, 208

pat\_trie\_.hpp, [4197](#)

pat\_trie\_base.hpp, [4197](#)

pbackfail

\_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2492](#)

\_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3507](#)

\_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3531](#)

std::basic\_filebuf< \_CharT, \_Traits >, [1395](#)

std::basic\_streambuf< \_CharT, \_Traits >, [1896](#)

std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2016](#)

std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3914](#)

pbase

\_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2493](#)

\_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3508](#)

\_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3531](#)

std::basic\_filebuf< \_CharT, \_Traits >, [1396](#)

std::basic\_streambuf< \_CharT, \_Traits >, [1897](#)

std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2017](#)

std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3914](#)

pbump

\_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2493](#)

\_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3508](#)

\_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3532](#)

std::basic\_filebuf< \_CharT, \_Traits >, [1396](#)

std::basic\_streambuf< \_CharT, \_Traits >, [1897](#)

std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2017](#)

std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3915](#)

peek

std::basic\_fstream< \_CharT, \_Traits >, [1456](#)

std::basic\_ifstream< \_CharT, \_Traits >, [1515](#)

std::basic\_iostream< \_CharT, \_Traits >, [1611](#)

std::basic\_istream< \_CharT, \_Traits >, [1666](#)

std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1719](#)

std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2074](#)

perms

Filesystem TS, [99](#)

piecewise\_construct

Utilities, [483](#)

pod\_char\_traits.h, [4198](#)

point\_const\_iterator.hpp, [4198](#), [4199](#)

point\_iterator.hpp, [4199](#)

point\_iterators.hpp, [4200](#)

pointer

\_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2124](#)

\_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2130](#)

\_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2802](#)

\_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, [2808](#)

std::allocator\_traits< \_Alloc >, [1308](#)

std::allocator\_traits< allocator< \_Tp > >, [1315](#)

std::back\_insert\_iterator< \_Container >, [1369](#)

std::front\_insert\_iterator< \_Container >, [2583](#)

std::insert\_iterator< \_Container >, [2699](#)

std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [2783](#)

std::istreambuf\_iterator< \_CharT, \_Traits >, [2788](#)

std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, [2794](#)

std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3198](#)

std::ostreambuf\_iterator< \_CharT, \_Traits >, [3203](#)

std::pointer\_traits< \_Ptr >, [3275](#)

std::pointer\_traits< \_Tp \* >, [3276](#)

std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3323](#)

std::set< \_Key, \_Compare, \_Alloc >, [3421](#)

std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3726](#)

std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3765](#)

std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3801](#)

std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3836](#)

Pointer Abstractions, [304](#)

\_\_cpp\_lib\_make\_unique, [307](#)

allocate\_shared, [307](#)

atomic\_compare\_exchange\_strong\_explicit, [308](#)

atomic\_exchange\_explicit, [308](#)

atomic\_is\_lock\_free, [309](#)

atomic\_load\_explicit, [309](#)

atomic\_store\_explicit, [310](#)

const\_pointer\_cast, [310](#)

dynamic\_pointer\_cast, [310](#)

get\_deleter, [311](#)

make\_shared, [311](#)

make\_unique, [311](#), [312](#)

operator!=, [312](#), [313](#)

operator<, [314](#), [315](#)

operator<<, [315](#)

operator<=, [315](#)–[317](#)

operator>, [318](#)–[320](#)

operator>=, [320](#), [321](#)

operator==, [317](#), [318](#)

static\_pointer\_cast, [321](#)

swap, [322](#)

Pointer Safety and Garbage Collection, [323](#)

declare\_no\_pointers, [324](#)

declare\_reachable, [324](#)

get\_pointer\_safety, [324](#)

pointer\_safety, [323](#)

- undeclare\_no\_pointers, [324](#)
- undeclare\_reachable, [324](#)
- pointer.h, [4200](#)
- pointer\_safety
  - Pointer Safety and Garbage Collection, [323](#)
- pointer\_to
  - std::pointer\_traits< \_Tp \* >, [3277](#)
- Poisson Distributions, [325](#)
  - operator!=, [326](#), [327](#)
  - operator<<, [328](#), [329](#)
  - operator>>, [329](#), [330](#)
- polar
  - Complex Numbers, [66](#)
- Policy-Based Data Structures, [332](#)
- policy\_access\_fnimps.hpp, [4202](#), [4203](#)
- pool\_allocator.h, [4203](#)
- pop
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3287](#)
  - std::queue< \_Tp, \_Sequence >, [3299](#)
  - std::stack< \_Tp, \_Sequence >, [3495](#)
- pop\_back
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1028](#)
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, [1207](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1960](#)
  - std::deque< \_Tp, \_Alloc >, [2432](#)
  - std::list< \_Tp, \_Alloc >, [2851](#)
  - std::vector< \_Tp, \_Alloc >, [3892](#)
- pop\_front
  - \_\_gnu\_parallel::RestrictedBoundedConcurrentQueue< \_Tp >, [1207](#)
  - std::deque< \_Tp, \_Alloc >, [2432](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2569](#)
  - std::list< \_Tp, \_Alloc >, [2851](#)
- pop\_heap
  - Heap, [118](#), [119](#)
- pos\_format
  - std::moneypunct< \_CharT, \_Intl >, [3001](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3012](#)
- pos\_type
  - std::basic\_ios< \_CharT, \_Traits >, [1544](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1891](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3909](#)
- position
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2946](#)
- positive\_sign
  - std::moneypunct< \_CharT, \_Intl >, [3001](#)
  - std::moneypunct\_byname< \_CharT, \_Intl >, [3013](#)
- postypes.h, [4204](#)
- pow
  - Complex Numbers, [66](#), [67](#)
- power
  - SGL, [399](#), [400](#)
- pptr
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2494](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3509](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3532](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1397](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1898](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2018](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3915](#)
- precision
  - std::basic\_fstream< \_CharT, \_Traits >, [1457](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1516](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1554](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1611](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1666](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1719](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1767](#), [1768](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1810](#), [1811](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1854](#), [1855](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2075](#)
  - std::ios\_base, [2712](#)
- predefined\_ops.h, [4204](#)
- prefix
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2947](#)
- prefix\_range
  - \_\_gnu\_pbds::trie\_prefix\_search\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, [3663](#), [3664](#)
- prefix\_search\_node\_update\_imp.hpp, [4205](#)
- prev\_permutation
  - Sorting, [434](#)
- priority\_queue
  - Heap-Based, [123](#)
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3285](#), [3286](#)
- priority\_queue.hpp, [4206](#)
- priority\_queue\_base\_dispatch.hpp, [4206](#)
- probabilities
  - std::discrete\_distribution< \_IntType >, [2454](#)
- probe\_fn\_base.hpp, [4206](#)
- propagate\_const, [4207](#)
- propagate\_on\_container\_copy\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [870](#)
  - std::allocator\_traits< \_Alloc >, [1308](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1315](#)
- propagate\_on\_container\_move\_assignment
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [870](#)
  - std::allocator\_traits< \_Alloc >, [1308](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1315](#)
- propagate\_on\_container\_swap



- `__gnu_cxx::__alloc_traits< _Alloc, typename >`, 871
- `std::allocator_traits< _Alloc >`, 1308
- `std::allocator_traits< allocator< _Tp > >`, 1315
- `ptr_fun`
  - Adaptors for pointers to functions, 4
- `ptr_traits.h`, 4209
- `pubimbue`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2494
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3509
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3532
  - `std::basic_filebuf< _CharT, _Traits >`, 1397
  - `std::basic_streambuf< _CharT, _Traits >`, 1898
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2018
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3915
- `pubseekoff`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2494
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3509
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3533
  - `std::basic_filebuf< _CharT, _Traits >`, 1397
  - `std::basic_streambuf< _CharT, _Traits >`, 1898
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2018
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3916
- `pubseekpos`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2495
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3510
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3533
  - `std::basic_filebuf< _CharT, _Traits >`, 1398
  - `std::basic_streambuf< _CharT, _Traits >`, 1899
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2019
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3916
- `pubsetbuf`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2495
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3510
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3535
  - `std::basic_filebuf< _CharT, _Traits >`, 1398
  - `std::basic_streambuf< _CharT, _Traits >`, 1899
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2019
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3917
- `pubsync`
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2496
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3511
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3535
  - `std::basic_filebuf< _CharT, _Traits >`, 1399
  - `std::basic_streambuf< _CharT, _Traits >`, 1900
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2020
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3917
- `push`
  - `std::priority_queue< _Tp, _Sequence, _Compare >`, 3287
- `std::queue< _Tp, _Sequence >`, 3300
- `std::stack< _Tp, _Sequence >`, 3495
- `push_back`
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1029
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1960
  - `std::deque< _Tp, _Alloc >`, 2432
  - `std::list< _Tp, _Alloc >`, 2851
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2478
  - `std::vector< _Tp, _Alloc >`, 3892
- `push_front`
  - `__gnu_parallel::__RestrictedBoundedConcurrentQueue< _Tp >`, 1207
  - `std::deque< _Tp, _Alloc >`, 2433
  - `std::forward_list< _Tp, _Alloc >`, 2570
  - `std::list< _Tp, _Alloc >`, 2852
- `push_heap`
  - Heap, 119, 120
- `put`
  - `std::basic_fstream< _CharT, _Traits >`, 1458
  - `std::basic_iostream< _CharT, _Traits >`, 1612
  - `std::basic_ofstream< _CharT, _Traits >`, 1768
  - `std::basic_ostream< _CharT, _Traits >`, 1811
  - `std::basic_ostringstream< _CharT, _Traits, _Alloc >`, 1855
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2076
  - `std::money_put< _CharT, _Outlter >`, 2987, 2988
  - `std::num_put< _CharT, _Outlter >`, 3141–3148
  - `std::time_put< _CharT, _Outlter >`, 3626, 3627
  - `std::time_put_byname< _CharT, _Outlter >`, 3631
- `put_money`
  - std, 778
- `put_time`
  - std, 778
- `putback`
  - `std::basic_fstream< _CharT, _Traits >`, 1458
  - `std::basic_ifstream< _CharT, _Traits >`, 1517
  - `std::basic_iostream< _CharT, _Traits >`, 1612
  - `std::basic_istream< _CharT, _Traits >`, 1667
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1720
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2076
- `pword`
  - `std::basic_fstream< _CharT, _Traits >`, 1459
  - `std::basic_ifstream< _CharT, _Traits >`, 1517
  - `std::basic_ios< _CharT, _Traits >`, 1554
  - `std::basic_iostream< _CharT, _Traits >`, 1613
  - `std::basic_istream< _CharT, _Traits >`, 1667
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1720
  - `std::basic_ofstream< _CharT, _Traits >`, 1769
  - `std::basic_ostream< _CharT, _Traits >`, 1812

- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1856
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2077
- std::ios\_base, 2713
- qsb\_steals
  - \_\_gnu\_parallel:: Settings, 1284
- quadratic\_probe\_fn\_imp.hpp, 4209
- queue, 4210
  - std::queue< \_Tp, \_Sequence >, 3298
- queue.h, 4210
  - \_GLIBCXX\_VOLATILE, 4210
- quicksort.h, 4211
- quiet\_NaN
  - std::numeric\_limits< \_Tp >, 3152
- quoted
  - std, 779
- quoted\_string.h, 4211
- r\_erase\_fn\_imps.hpp, 4212
- radix
  - std::\_\_numeric\_limits\_base, 953
  - std::numeric\_limits< \_Tp >, 3156
- random, 4212
- Random Number Distributions, 333
- Random Number Generation, 334
  - generate\_canonical, 334
- Random Number Generators, 335
  - minstd\_rand, 336
  - minstd\_rand0, 336
  - mt19937, 336
  - mt19937\_64, 337
  - operator!=, 337–339
  - operator<<, 340
- Random Number Utilities, 341
- random.h, 4213
- random.tcc, 4217, 4221
- random\_number.h, 4223
- random\_sample
  - SGL, 400
- random\_sample\_n
  - SGL, 401
- random\_shuffle
  - Mutating, 208
- random\_shuffle.h, 4223
- random\_shuffle\_minimal\_n
  - \_\_gnu\_parallel:: Settings, 1284
- range\_access.h, 4224
- range\_cmp.h, 4226
- ranged\_hash\_fn.hpp, 4226
- ranged\_probe\_fn.hpp, 4226
- ranges, 4227
- ranges\_algo.h, 4227
- ranges\_algobase.h, 4227
- ranges\_uninitialized.h, 4228
- ratio, 4228, 4229
- ratio\_add
  - Rational Arithmetic, 343
- ratio\_divide
  - Rational Arithmetic, 343
- ratio\_multiply
  - Rational Arithmetic, 343
- ratio\_subtract
  - Rational Arithmetic, 343
- Rational Arithmetic, 342
  - ratio\_add, 343
  - ratio\_divide, 343
  - ratio\_multiply, 343
  - ratio\_subtract, 343
- rb\_tree, 4230
- rb\_tree.hpp, 4230
- rbegin
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1029
- std, 779, 780
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1961
  - std::deque< \_Tp, \_Alloc >, 2433
  - std::list< \_Tp, \_Alloc >, 2852
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2930
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3048
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3076
  - std::set< \_Key, \_Compare, \_Alloc >, 3442
  - std::vector< \_Tp, \_Alloc >, 3893
- rc.hpp, 4231
- rc\_binomial\_heap.hpp, 4231
- rc\_string\_base.h, 4231
- rdbuf
  - std::basic\_fstream< \_CharT, \_Traits >, 1459, 1460
  - std::basic\_ifstream< \_CharT, \_Traits >, 1518
  - std::basic\_ios< \_CharT, \_Traits >, 1555
  - std::basic\_iostream< \_CharT, \_Traits >, 1613, 1614
  - std::basic\_istream< \_CharT, \_Traits >, 1668
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1721
  - std::basic\_ofstream< \_CharT, \_Traits >, 1769, 1770
  - std::basic\_ostream< \_CharT, \_Traits >, 1812, 1813
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1856, 1857
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2077, 2078
- rdstate
  - std::basic\_fstream< \_CharT, \_Traits >, 1460
  - std::basic\_ifstream< \_CharT, \_Traits >, 1518
  - std::basic\_ios< \_CharT, \_Traits >, 1556
  - std::basic\_iostream< \_CharT, \_Traits >, 1614
  - std::basic\_istream< \_CharT, \_Traits >, 1669



- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1722
- std::basic\_ofstream< \_CharT, \_Traits >, 1770
- std::basic\_ostream< \_CharT, \_Traits >, 1813
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1857
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2078
- read
  - std::basic\_fstream< \_CharT, \_Traits >, 1460
  - std::basic\_ifstream< \_CharT, \_Traits >, 1519
  - std::basic\_iostream< \_CharT, \_Traits >, 1615
  - std::basic\_istream< \_CharT, \_Traits >, 1669
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1722
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2078
- readsome
  - std::basic\_fstream< \_CharT, \_Traits >, 1461
  - std::basic\_ifstream< \_CharT, \_Traits >, 1519
  - std::basic\_iostream< \_CharT, \_Traits >, 1615
  - std::basic\_istream< \_CharT, \_Traits >, 1670
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1723
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2079
- ready
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2947
- rebind
  - std::pointer\_traits< \_Ptr >, 3275
- ref
  - std::reference\_wrapper< \_Tp >, 3343
- reference
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 2106
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, 2113
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2125
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, 2130
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, 2802
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_const\_iterator< Node, \_Alloc >, 2808
  - std::back\_insert\_iterator< \_Container >, 1370
  - std::front\_insert\_iterator< \_Container >, 2583
  - std::insert\_iterator< \_Container >, 2699
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2784
  - std::istreambuf\_iterator< \_CharT, \_Traits >, 2788
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2795
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3198
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, 3203
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, 3323
  - std::set< \_Key, \_Compare, \_Alloc >, 3422
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3726
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3765
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3801
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3836
  - refwrap.h, 4232
  - regex, 4232
    - Regular Expressions, 349
  - regex.h, 4233
  - regex.tcc, 4235
  - regex\_automaton.h, 4236
  - regex\_automaton.tcc, 4237
  - regex\_compiler.h, 4237
  - regex\_compiler.tcc, 4238
  - regex\_constants.h, 4238
  - regex\_error
    - std::regex\_error, 3344
  - regex\_error.h, 4240
  - regex\_executor.h, 4241
  - regex\_executor.tcc, 4241
  - regex\_iterator
    - std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3346, 3347
  - regex\_match
    - Regular Expressions, 376–380
  - regex\_replace
    - Regular Expressions, 380–384
  - regex\_scanner.h, 4241
  - regex\_scanner.tcc, 4242
  - regex\_search
    - Regular Expressions, 384–389
  - regex\_token\_iterator
    - std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, 3350–3352
  - regex\_traits
    - std::regex\_traits< \_Ch\_type >, 3356
  - register\_callback
    - std::basic\_fstream< \_CharT, \_Traits >, 1462
    - std::basic\_ifstream< \_CharT, \_Traits >, 1520
    - std::basic\_ios< \_CharT, \_Traits >, 1556
    - std::basic\_iostream< \_CharT, \_Traits >, 1616
    - std::basic\_istream< \_CharT, \_Traits >, 1671
    - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1724
    - std::basic\_ofstream< \_CharT, \_Traits >, 1770
    - std::basic\_ostream< \_CharT, \_Traits >, 1814

- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1857
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2080
- std::ios\_base, 2713
- Regular Expressions, 344
  - cregex\_token\_iterator, 349
  - csub\_match, 349
  - operator!=, 351, 352, 354, 355
  - operator<, 355–358
  - operator<<, 358
  - operator<=, 359, 361–363
  - operator>, 367, 368, 370, 371
  - operator>=, 372, 373, 375
  - operator==, 363, 364, 366, 367
  - regex, 349
  - regex\_match, 376–380
  - regex\_replace, 380–384
  - regex\_search, 384–389
  - sregex\_token\_iterator, 349
  - ssub\_match, 350
  - swap, 389, 390
  - wcregex\_token\_iterator, 350
  - wcsub\_match, 350
  - wregex, 350
  - wsregex\_token\_iterator, 350
  - wssub\_match, 351
- rehash
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3751
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3787
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3823
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3858
- release
  - std::auto\_ptr< \_Tp >, 1366
  - std::unique\_ptr< \_Tp, \_Dp >, 3711
  - std::unique\_ptr< \_Tp[], \_Dp >, 3718
- remove
  - Mutating, 209
  - std::forward\_list< \_Tp, \_Alloc >, 2570
  - std::list< \_Tp, \_Alloc >, 2852
- remove\_all\_extents\_t
  - Metaprogramming, 195
- remove\_const\_t
  - Metaprogramming, 195
- remove\_copy
  - Mutating, 210
- remove\_copy\_if
  - Mutating, 210
- remove\_cv\_t
  - Metaprogramming, 195
- remove\_extent\_t
  - Metaprogramming, 196
- remove\_if
  - Mutating, 211
  - std::forward\_list< \_Tp, \_Alloc >, 2571
  - std::list< \_Tp, \_Alloc >, 2853
- remove\_pointer\_t
  - Metaprogramming, 196
- remove\_reference\_t
  - Metaprogramming, 196
- remove\_volatile\_t
  - Metaprogramming, 196
- rend
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1030
  - std, 780, 781
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1961
  - std::deque< \_Tp, \_Alloc >, 2434
  - std::list< \_Tp, \_Alloc >, 2853
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2930, 2931
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3048
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3077
  - std::set< \_Key, \_Compare, \_Alloc >, 3443
  - std::vector< \_Tp, \_Alloc >, 3893
- replace
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1030–1038
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 1998–2003
  - Mutating, 211
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1961–1969
- replace\_copy
  - std, 782
- replace\_copy\_if
  - Mutating, 212
- replace\_if
  - Mutating, 213
- replace\_minimal\_n
  - \_\_gnu\_parallel::\_Settings, 1285
- requested\_size
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3579
  - std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1293
- reserve
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1039
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 2004
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1969

- `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3751
- `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3787
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3823
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3858
- `std::vector< _Tp, _Alloc >`, 3894
- reset
  - `std::auto_ptr< _Tp >`, 1366
  - `std::bernoulli_distribution`, 2101
  - `std::binomial_distribution< _IntType >`, 2141
  - `std::bitset< _Nb >`, 2163
  - `std::cauchy_distribution< _RealType >`, 2175
  - `std::chi_squared_distribution< _RealType >`, 2202
  - `std::discrete_distribution< _IntType >`, 2454
  - `std::exponential_distribution< _RealType >`, 2528
  - `std::extreme_value_distribution< _RealType >`, 2533
  - `std::fisher_f_distribution< _RealType >`, 2542
  - `std::gamma_distribution< _RealType >`, 2608
  - `std::geometric_distribution< _IntType >`, 2612
  - `std::lognormal_distribution< _RealType >`, 2885
  - `std::negative_binomial_distribution< _IntType >`, 3096
  - `std::normal_distribution< _RealType >`, 3105
  - `std::piecewise_constant_distribution< _RealType >`, 3261
  - `std::piecewise_linear_distribution< _RealType >`, 3267
  - `std::poisson_distribution< _IntType >`, 3281
  - `std::student_t_distribution< _RealType >`, 3554
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2478
  - `std::uniform_int_distribution< _IntType >`, 3698
  - `std::uniform_real_distribution< _RealType >`, 3702
  - `std::unique_ptr< _Tp, _Dp >`, 3712
  - `std::unique_ptr< _Tp[], _Dp >`, 3718
  - `std::weibull_distribution< _RealType >`, 3932
- resetsiosflags
  - `std`, 782
- resize
  - `__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >`, 1039, 1040
  - `__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >`, 2681
  - Numeric Arrays, 299
  - `std::basic_string< _CharT, _Traits, _Alloc >`, 1970, 1971
  - `std::deque< _Tp, _Alloc >`, 2434, 2435
  - `std::forward_list< _Tp, _Alloc >`, 2571, 2572
  - `std::list< _Tp, _Alloc >`, 2854
  - `std::tr2::dynamic_bitset< _WordT, _Alloc >`, 2479
  - `std::vector< _Tp, _Alloc >`, 3894, 3895
  - `resize_fn_imps.hpp`, 4242
  - `resize_no_store_hash_fn_imps.hpp`, 4242
  - `resize_policy.hpp`, 4242
  - `resize_store_hash_fn_imps.hpp`, 4243
  - result\_of\_t
    - Metaprogramming, 196
  - result\_type
    - `__gnu_cxx::__detail::__Ffit_finder< _Tp >`, 1077
    - `__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >`, 2118
    - `__gnu_cxx::project1st< _Arg1, _Arg2 >`, 3290
    - `__gnu_cxx::project2nd< _Arg1, _Arg2 >`, 3291
    - `__gnu_cxx::select1st< _Pair >`, 3409
    - `__gnu_cxx::select2nd< _Pair >`, 3410
    - `__gnu_cxx::subtractive_rng`, 3573
    - `__gnu_cxx::unary_compose< _Operation1, _Operation2 >`, 3689
    - `__gnu_parallel::EqualFromLess< _T1, _T2, _Compare >`, 1069
    - `__gnu_parallel::EqualTo< _T1, _T2 >`, 1073
    - `__gnu_parallel::Less< _T1, _T2 >`, 1128
    - `__gnu_parallel::Lexicographic< _T1, _T2, _Compare >`, 1130
    - `__gnu_parallel::LexicographicReverse< _T1, _T2, _Compare >`, 1132
    - `__gnu_parallel::Multiplies< _Tp1, _Tp2, _Result >`, 1172
    - `__gnu_parallel::Plus< _Tp1, _Tp2, _Result >`, 1188
    - `__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 887
    - `__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >`, 889
    - `__gnu_parallel::__unary_negate< _Predicate, argument_type >`, 975
  - `std::bernoulli_distribution`, 2099
  - `std::binary_function< _Arg1, _Arg2, _Result >`, 2119
  - `std::binary_negate< _Predicate >`, 2134
  - `std::binder1st< _Operation >`, 2136
  - `std::binder2nd< _Operation >`, 2137
  - `std::binomial_distribution< _IntType >`, 2139
  - `std::cauchy_distribution< _RealType >`, 2173
  - `std::chi_squared_distribution< _RealType >`, 2200
  - `std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >`, 2269
  - `std::const_mem_fun1_t< _Ret, _Tp, _Arg >`, 2271
  - `std::const_mem_fun_ref_t< _Ret, _Tp >`, 2273
  - `std::const_mem_fun_t< _Ret, _Tp >`, 2274
  - `std::discard_block_engine< _RandomNumberEngine, __p, __r >`, 2446
  - `std::discrete_distribution< _IntType >`, 2453
  - `std::divides< _Tp >`, 2457
  - `std::equal_to< _Tp >`, 2516

- std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, [3219](#)
- std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, [3224](#)
- std::exponential\_distribution< \_RealType >, [2525](#)
- std::extreme\_value\_distribution< \_RealType >, [2530](#)
- std::fisher\_f\_distribution< \_RealType >, [2540](#)
- std::gamma\_distribution< \_RealType >, [2605](#)
- std::geometric\_distribution< \_IntType >, [2610](#)
- std::greater< \_Tp >, [2627](#)
- std::greater\_equal< \_Tp >, [2630](#)
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, [2640](#)
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, [2642](#)
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, [2686](#)
- std::less< \_Tp >, [2812](#)
- std::less\_equal< \_Tp >, [2815](#)
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, [2818](#)
- std::logical\_and< \_Tp >, [2877](#)
- std::logical\_not< \_Tp >, [2879](#)
- std::logical\_or< \_Tp >, [2881](#)
- std::lognormal\_distribution< \_RealType >, [2883](#)
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, [2952](#)
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, [2954](#)
- std::mem\_fun\_ref\_t< \_Ret, \_Tp >, [2955](#)
- std::mem\_fun\_t< \_Ret, \_Tp >, [2957](#)
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2959](#)
- std::minus< \_Tp >, [2971](#)
- std::modulus< \_Tp >, [2975](#)
- std::multiplies< \_Tp >, [3053](#)
- std::negate< \_Tp >, [3091](#)
- std::negative\_binomial\_distribution< \_IntType >, [3094](#)
- std::normal\_distribution< \_RealType >, [3102](#)
- std::not\_equal\_to< \_Tp >, [3108](#)
- std::owner\_less< shared\_ptr< \_Tp > >, [3220](#)
- std::owner\_less< void >, [3222](#)
- std::owner\_less< weak\_ptr< \_Tp > >, [3223](#)
- std::piecewise\_constant\_distribution< \_RealType >, [3259](#)
- std::piecewise\_linear\_distribution< \_RealType >, [3265](#)
- std::plus< \_Tp >, [3270](#)
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, [3272](#)
- std::pointer\_to\_unary\_function< \_Arg, \_Result >, [3274](#)
- std::poisson\_distribution< \_IntType >, [3279](#)
- std::random\_device, [3305](#)
- std::seed\_seq, [3407](#)
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, [3477](#)
- std::student\_t\_distribution< \_RealType >, [3553](#)
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3567](#)
- std::unary\_function< \_Arg, \_Result >, [3690](#)
- std::unary\_negate< \_Predicate >, [3692](#)
- std::uniform\_int\_distribution< \_IntType >, [3696](#)
- std::uniform\_real\_distribution< \_RealType >, [3700](#)
- std::weibull\_distribution< \_RealType >, [3929](#)
- rethrow\_exception
  - Exceptions, [90](#)
- rethrow\_if\_nested
  - Exceptions, [90](#)
- return\_temporary\_buffer
  - std, [783](#)
- reverse
  - Mutating, [213](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2572](#)
  - std::list< \_Tp, \_Alloc >, [2854](#)
- reverse\_copy
  - Mutating, [214](#)
- reverse\_iteration
  - \_\_gnu\_pbds::container\_traits< Cntnr >, [2294](#)
- reverse\_iterator
  - std::reverse\_iterator< \_Iterator >, [3371](#), [3372](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3422](#)
- rfind
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1040–1042](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [2004](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1971–1973](#)
- riemann\_zeta
  - Mathematical Special Functions, [178](#)
  - TR1 Mathematical Special Functions, [447](#)
- riemann\_zetaf
  - Mathematical Special Functions, [178](#)
- riemann\_zetal
  - Mathematical Special Functions, [179](#)
- right
  - std, [783](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1477](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1533](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1566](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1631](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1684](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1738](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1783](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1826](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1870](#)

- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2095
- std::ios\_base, 2722
- rope, 4243
- ropeimpl.h, 4246
- rotate
  - Mutating, 214
- rotate\_copy
  - Mutating, 215
- rotate\_fn\_imps.hpp, 4247
- round\_error
  - std::numeric\_limits< \_Tp >, 3152
- round\_style
  - std::\_\_numeric\_limits\_base, 953
  - std::numeric\_limits< \_Tp >, 3157
- round\_to\_nearest
  - std, 691
- round\_toward\_infinity
  - std, 691
- round\_toward\_neg\_infinity
  - std, 691
- round\_toward\_zero
  - std, 691
- runtime\_error
  - std::runtime\_error, 3382
- safe\_base.h, 4247
- safe\_container.h, 4248
- safe\_iterator.h, 4248
- safe\_iterator.tcc, 4249
- safe\_local\_iterator.h, 4250
- safe\_local\_iterator.tcc, 4251
- safe\_sequence.h, 4251
- safe\_sequence.tcc, 4251
- safe\_unordered\_base.h, 4252
- safe\_unordered\_container.h, 4252
- safe\_unordered\_container.tcc, 4252
- sample
  - experimental/algorithm, 3953
- sample\_probe\_fn
  - \_\_gnu\_pbds::sample\_probe\_fn, 3383
- sample\_probe\_fn.hpp, 4253
- sample\_range\_hashing
  - \_\_gnu\_pbds::sample\_range\_hashing, 3385
  - \_\_gnu\_pbds::sample\_resize\_policy, 3392
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3396
  - \_\_gnu\_pbds::sample\_size\_policy, 3398
- sample\_range\_hashing.hpp, 4253
- sample\_ranged\_hash\_fn
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, 3386, 3387
- sample\_ranged\_hash\_fn.hpp, 4253
- sample\_ranged\_probe\_fn.hpp, 4254
- sample\_resize\_policy
  - \_\_gnu\_pbds::sample\_resize\_policy, 3389
- sample\_resize\_policy.hpp, 4254
- sample\_resize\_trigger
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3394
- sample\_resize\_trigger.hpp, 4254
- sample\_size\_policy
  - \_\_gnu\_pbds::sample\_size\_policy, 3398
- sample\_size\_policy.hpp, 4255
- sample\_tree\_node\_update.hpp, 4255
- sample\_trie\_access\_traits.hpp, 4255
- sample\_trie\_node\_update
  - \_\_gnu\_pbds::sample\_trie\_node\_update< Node\_Cltr, Node\_Itr, \_ATraits, \_Alloc >, 3401
- sample\_trie\_node\_update.hpp, 4256
- sample\_update\_policy
  - \_\_gnu\_pbds::sample\_update\_policy, 3402
- sample\_update\_policy.hpp, 4256
- sbumpc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2496
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3511
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3535
  - std::basic\_filebuf< \_CharT, \_Traits >, 1399
  - std::basic\_streambuf< \_CharT, \_Traits >, 1900
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2020
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3917
- scan\_is
  - std::\_\_ctype\_abstract\_base< \_CharT >, 911
  - std::ctype< \_CharT >, 2313
  - std::ctype< char >, 2330
  - std::ctype< wchar\_t >, 2349
  - std::ctype\_byname< \_CharT >, 2367
  - std::ctype\_byname< char >, 2383
- scan\_not
  - std::\_\_ctype\_abstract\_base< \_CharT >, 911
  - std::ctype< \_CharT >, 2314
  - std::ctype< char >, 2330
  - std::ctype< wchar\_t >, 2350
  - std::ctype\_byname< \_CharT >, 2367
  - std::ctype\_byname< char >, 2383
- scientific
  - std, 783
  - std::basic\_fstream< \_CharT, \_Traits >, 1478
  - std::basic\_ifstream< \_CharT, \_Traits >, 1534
  - std::basic\_ios< \_CharT, \_Traits >, 1567
  - std::basic\_iostream< \_CharT, \_Traits >, 1632
  - std::basic\_istream< \_CharT, \_Traits >, 1684
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1738
  - std::basic\_ofstream< \_CharT, \_Traits >, 1783
  - std::basic\_ostream< \_CharT, \_Traits >, 1826
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, 1871
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2096

- std::ios\_base, [2722](#)
- scoped\_allocator, [4256](#)
- search
  - Non-Mutating, [247](#), [248](#)
- search.h, [4257](#)
- search\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1285](#)
- search\_n
  - Non-Mutating, [248](#), [249](#)
- second
  - \_\_gnu\_parallel::IteratorPair<\_Iterator1, \_Iterator2, \_IteratorCategory>, [1123](#)
  - std::pair<\_T1, \_T2>, [3230](#)
  - std::sub\_match<\_Bilter>, [3566](#)
- second\_argument\_type
  - \_\_gnu\_cxx::project1st<\_Arg1, \_Arg2>, [3290](#)
  - \_\_gnu\_cxx::project2nd<\_Arg1, \_Arg2>, [3291](#)
  - \_\_gnu\_parallel::EqualFromLess<\_T1, \_T2, \_Compare>, [1069](#)
  - \_\_gnu\_parallel::EqualTo<\_T1, \_T2>, [1073](#)
  - \_\_gnu\_parallel::Less<\_T1, \_T2>, [1128](#)
  - \_\_gnu\_parallel::Lexicographic<\_T1, \_T2, \_Compare>, [1130](#)
  - \_\_gnu\_parallel::LexicographicReverse<\_T1, \_T2, \_Compare>, [1132](#)
  - \_\_gnu\_parallel::Multiplies<\_Tp1, \_Tp2, \_Result>, [1172](#)
  - \_\_gnu\_parallel::Plus<\_Tp1, \_Tp2, \_Result>, [1188](#)
  - std::binary\_function<\_Arg1, \_Arg2, \_Result>, [2120](#)
  - std::binary\_negate<\_Predicate>, [2134](#)
  - std::const\_mem\_fun1\_ref\_t<\_Ret, \_Tp, \_Arg>, [2269](#)
  - std::const\_mem\_fun1\_t<\_Ret, \_Tp, \_Arg>, [2271](#)
  - std::divides<\_Tp>, [2457](#)
  - std::equal\_to<\_Tp>, [2516](#)
  - std::experimental::fundamentals\_v2::owner\_less<shared\_ptr<\_Tp>>, [3219](#)
  - std::experimental::fundamentals\_v2::owner\_less<weak\_ptr<\_Tp>>, [3224](#)
  - std::greater<\_Tp>, [2627](#)
  - std::greater\_equal<\_Tp>, [2630](#)
  - std::less<\_Tp>, [2812](#)
  - std::less\_equal<\_Tp>, [2815](#)
  - std::logical\_and<\_Tp>, [2877](#)
  - std::logical\_or<\_Tp>, [2881](#)
  - std::mem\_fun1\_ref\_t<\_Ret, \_Tp, \_Arg>, [2952](#)
  - std::mem\_fun1\_t<\_Ret, \_Tp, \_Arg>, [2954](#)
  - std::minus<\_Tp>, [2971](#)
  - std::modulus<\_Tp>, [2975](#)
  - std::multiplies<\_Tp>, [3053](#)
  - std::not\_equal\_to<\_Tp>, [3108](#)
  - std::owner\_less<shared\_ptr<\_Tp>>, [3220](#)
  - std::owner\_less<void>, [3222](#)
  - std::owner\_less<weak\_ptr<\_Tp>>, [3223](#)
  - std::plus<\_Tp>, [3270](#)
  - std::pointer\_to\_binary\_function<\_Arg1, \_Arg2, \_Result>, [3272](#)
  - second\_type
    - \_\_gnu\_parallel::IteratorPair<\_Iterator1, \_Iterator2, \_IteratorCategory>, [1120](#)
    - std::pair<\_T1, \_T2>, [3228](#)
    - std::sub\_match<\_Bilter>, [3560](#)
  - seconds
    - Time, [454](#)
  - seed
    - std::discard\_block\_engine<\_RandomNumberEngine, \_\_p, \_\_r>, [2449](#)
    - std::independent\_bits\_engine<\_RandomNumberEngine, \_\_w, \_UIntType>, [2689](#)
    - std::linear\_congruential\_engine<\_UIntType, \_\_a, \_\_c, \_\_m>, [2821](#)
    - std::shuffle\_order\_engine<\_RandomNumberEngine, \_\_k>, [3480](#)
    - std::subtract\_with\_carry\_engine<\_UIntType, \_\_w, \_\_s, \_\_r>, [3569](#), [3570](#)
  - seed\_seq
    - std::seed\_seq, [3408](#)
  - seekdir
    - std::basic\_fstream<\_CharT, \_Traits>, [1422](#)
    - std::basic\_ifstream<\_CharT, \_Traits>, [1490](#)
    - std::basic\_ios<\_CharT, \_Traits>, [1544](#)
    - std::basic\_iostream<\_CharT, \_Traits>, [1578](#)
    - std::basic\_istream<\_CharT, \_Traits>, [1642](#)
    - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1694](#)
    - std::basic\_ofstream<\_CharT, \_Traits>, [1747](#)
    - std::basic\_ostream<\_CharT, \_Traits>, [1793](#)
    - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1836](#)
    - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2041](#)
    - std::ios\_base, [2709](#)
  - seekg
    - std::basic\_fstream<\_CharT, \_Traits>, [1462](#), [1463](#)
    - std::basic\_ifstream<\_CharT, \_Traits>, [1521](#)
    - std::basic\_iostream<\_CharT, \_Traits>, [1617](#)
    - std::basic\_istream<\_CharT, \_Traits>, [1671](#), [1672](#)
    - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1724](#), [1725](#)
    - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2080](#), [2081](#)
  - seekoff
    - \_\_gnu\_cxx::enc\_filebuf<\_CharT>, [2496](#)
    - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, [3511](#)
    - \_\_gnu\_cxx::stdio\_sync\_filebuf<\_CharT, \_Traits>, [3535](#)
    - std::basic\_filebuf<\_CharT, \_Traits>, [1399](#)
    - std::basic\_streambuf<\_CharT, \_Traits>, [1900](#)



- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2020](#)
- std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3917](#)
- seekp
  - std::basic\_fstream< \_CharT, \_Traits >, [1463](#), [1464](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1618](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1771](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1814](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1858](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2081](#), [2082](#)
- seekpos
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2496](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3511](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3536](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1399](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1900](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2020](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3918](#)
- select\_on\_container\_copy\_construction
  - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [877](#)
  - std::allocator\_traits< \_Alloc >, [1312](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1319](#)
- sentry
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [3412](#)
  - std::basic\_ostream< \_CharT, \_Traits >::sentry, [3414](#)
- Sequences, [403](#)
- sequential
  - \_\_gnu\_parallel, [534](#)
- set, [4257](#), [4258](#)
  - \_\_gnu\_parallel::Settings, [1278](#)
  - std::bitset< \_Nb >, [2163](#), [2164](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3423–3426](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2479](#)
- Set Operations, [404](#)
  - includes, [405](#)
  - set\_difference, [406](#)
  - set\_intersection, [407](#), [408](#)
  - set\_symmetric\_difference, [409](#)
  - set\_union, [411](#), [412](#)
- set.h, [4259](#)
- set\_default\_resource
  - experimental/memory\_resource, [4161](#)
- set\_difference
  - Set Operations, [406](#)
- set\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1285](#)
- set\_intersection
  - Set Operations, [407](#), [408](#)
- set\_intersection\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1285](#)
- set\_load
  - \_\_gnu\_pbds::cc\_hash\_max\_collision\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2181](#)
- set\_loads
  - \_\_gnu\_pbds::hash\_load\_check\_resize\_trigger< External\_Load\_Access, Size\_Type >, [2668](#)
- set\_new\_handler
  - std, [784](#)
- set\_num\_threads
  - \_\_gnu\_parallel::balanced\_quicksort\_tag, [1381](#)
  - \_\_gnu\_parallel::balanced\_tag, [1382](#)
  - \_\_gnu\_parallel::default\_parallel\_tag, [2401](#)
  - \_\_gnu\_parallel::exact\_tag, [2521](#)
  - \_\_gnu\_parallel::multiway\_mergesort\_exact\_tag, [3086](#)
  - \_\_gnu\_parallel::multiway\_mergesort\_sampling\_tag, [3088](#)
  - \_\_gnu\_parallel::multiway\_mergesort\_tag, [3089](#)
  - \_\_gnu\_parallel::omp\_loop\_static\_tag, [3191](#)
  - \_\_gnu\_parallel::omp\_loop\_tag, [3192](#)
  - \_\_gnu\_parallel::parallel\_tag, [3235](#)
  - \_\_gnu\_parallel::quicksort\_tag, [3302](#)
  - \_\_gnu\_parallel::sampling\_tag, [3404](#)
  - \_\_gnu\_parallel::unbalanced\_tag, [3693](#)
- set\_operations.h, [4260](#)
- set\_symmetric\_difference
  - Set Operations, [409](#)
- set\_symmetric\_difference\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1285](#)
- set\_terminate
  - Exceptions, [90](#)
- set\_unexpected
  - Exceptions, [90](#)
- set\_union
  - Set Operations, [411](#), [412](#)
- set\_union\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1286](#)
- setbase
  - std, [784](#)
- setbuf
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2497](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3512](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3536](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1400](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1901](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2021](#)
  - std::wbuffer\_convert< \_Codecv, \_Elem, \_Tr >, [3918](#)
- setf
  - std::basic\_fstream< \_CharT, \_Traits >, [1464](#), [1465](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1522](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1557](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1619](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1672](#), [1673](#)

- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, [1725](#), [1726](#)
- std::basic\_ofstream< \_CharT, \_Traits >, [1772](#)
- std::basic\_ostream< \_CharT, \_Traits >, [1815](#)
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1859](#)
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2082](#), [2083](#)
- std::ios\_base, [2714](#)
- setfill
  - std, [784](#)
- setg
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2497](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3512](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3537](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1400](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1901](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2021](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3918](#)
- setiosflags
  - std, [785](#)
- setp
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2498](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3513](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3537](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1401](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1902](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2022](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3919](#)
- setprecision
  - std, [785](#)
- setstate
  - std::basic\_fstream< \_CharT, \_Traits >, [1465](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1523](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1558](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1620](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1673](#)
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, [1726](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1773](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1816](#)
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, [1860](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2083](#)
- settings.h, [4260](#)
- \_GLIBCXX\_PARALLEL\_CONDITION, [4261](#)
- setw
  - std, [785](#)
- sgetc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2498](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3513](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3538](#)
- std::basic\_filebuf< \_CharT, \_Traits >, [1401](#)
- std::basic\_streambuf< \_CharT, \_Traits >, [1902](#)
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2022](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3919](#)
- sgetc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2499](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3514](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3538](#)
- std::basic\_filebuf< \_CharT, \_Traits >, [1402](#)
- std::basic\_streambuf< \_CharT, \_Traits >, [1902](#)
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2023](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3920](#)
- SGL, [391](#)
- \_Find\_first, [394](#)
- \_Find\_next, [395](#)
- \_Unchecked\_flip, [395](#)
- \_Unchecked\_reset, [396](#)
- \_Unchecked\_set, [396](#)
- \_Unchecked\_test, [396](#)
- \_\_median, [393](#), [394](#)
- compose1, [396](#)
- compose2, [397](#)
- constant0, [397](#)
- constant1, [397](#)
- constant2, [397](#)
- copy\_n, [398](#)
- distance, [398](#)
- identity\_element, [398](#), [399](#)
- lexicographical\_compare\_3way, [399](#)
- power, [399](#), [400](#)
- random\_sample, [400](#)
- random\_sample\_n, [401](#)
- uninitialized\_copy\_n, [401](#)
- shared\_future
  - std::shared\_future< \_Res >, [3454](#)
  - std::shared\_future< \_Res & >, [3457](#)
  - std::shared\_future< void >, [3460](#)
- shared\_mutex, [4262](#)
- shared\_ptr
  - std::shared\_ptr< \_Tp >, [3466](#)–[3473](#)
- shared\_ptr.h, [4262](#), [4263](#)
- shared\_ptr\_atomic.h, [4266](#)
- shared\_ptr\_base.h, [4266](#)
- shift
  - Numeric Arrays, [299](#)
- showbase
  - std, [786](#)
  - std::basic\_fstream< \_CharT, \_Traits >, [1478](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1534](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1567](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1632](#)



- std::basic\_istream< \_CharT, \_Traits >, 1684
- std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1738
- std::basic\_ofstream< \_CharT, \_Traits >, 1783
- std::basic\_ostream< \_CharT, \_Traits >, 1826
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1871
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2096
- std::ios\_base, 2722
- showmanyc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2499
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3514
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3539
- std::basic\_filebuf< \_CharT, \_Traits >, 1402
- std::basic\_streambuf< \_CharT, \_Traits >, 1903
- std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2023
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3920
- showpoint
  - std, 786
  - std::basic\_fstream< \_CharT, \_Traits >, 1478
  - std::basic\_ifstream< \_CharT, \_Traits >, 1534
  - std::basic\_ios< \_CharT, \_Traits >, 1567
  - std::basic\_iostream< \_CharT, \_Traits >, 1632
  - std::basic\_istream< \_CharT, \_Traits >, 1684
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1738
  - std::basic\_ofstream< \_CharT, \_Traits >, 1783
  - std::basic\_ostream< \_CharT, \_Traits >, 1827
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1871
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2096
  - std::ios\_base, 2723
- showpos
  - std, 786
  - std::basic\_fstream< \_CharT, \_Traits >, 1478
  - std::basic\_ifstream< \_CharT, \_Traits >, 1534
  - std::basic\_ios< \_CharT, \_Traits >, 1567
  - std::basic\_iostream< \_CharT, \_Traits >, 1632
  - std::basic\_istream< \_CharT, \_Traits >, 1685
  - std::basic\_istream< \_CharT, \_Traits, \_Alloc >, 1739
  - std::basic\_ofstream< \_CharT, \_Traits >, 1784
  - std::basic\_ostream< \_CharT, \_Traits >, 1827
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1871
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2096
  - std::ios\_base, 2723
- shrink\_to\_fit
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1042
- std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1973
- std::deque< \_Tp, \_Alloc >, 2435
- std::vector< \_Tp, \_Alloc >, 3895
- shuffle
  - Mutating, 216
- shuffle\_order\_engine
  - std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3477, 3478
- signaling\_NaN
  - std::numeric\_limits< \_Tp >, 3152
- sin
  - Complex Numbers, 68
- sinh
  - Complex Numbers, 68
- size
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, 1043
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3579
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, 2005
  - Numeric Arrays, 300
  - std::\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1293
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, 1973
  - std::bitset< \_Nb >, 2164
  - std::deque< \_Tp, \_Alloc >, 2435
  - std::list< \_Tp, \_Alloc >, 2855
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2931
  - std::match\_results< \_Bi\_iter, \_Alloc >, 2948
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3049
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3077
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, 3287
  - std::queue< \_Tp, \_Sequence >, 3301
  - std::set< \_Key, \_Compare, \_Alloc >, 3443
  - std::stack< \_Tp, \_Sequence >, 3495
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2480
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3752
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3787
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3824
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3858
  - std::vector< \_Tp, \_Alloc >, 3895
- size\_fn\_imps.hpp, 4268
- size\_type
  - \_\_gnu\_pbds::hash\_prime\_size\_policy, 2675
  - \_\_gnu\_pbds::sample\_range\_hashing, 3385
  - \_\_gnu\_pbds::sample\_resize\_policy, 3389
  - \_\_gnu\_pbds::sample\_resize\_trigger, 3393

- `__gnu_pbds::sample_size_policy`, 3397
- `__gnu_pbds::trie_prefix_search_node_update< Node_Cltr, Node_Itr, _ATraits, _Alloc >`, 3662
- `std::allocator_traits< _Alloc >`, 1309
- `std::allocator_traits< allocator< _Tp > >`, 1316
- `std::set< _Key, _Compare, _Alloc >`, 3422
- `std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3726
- `std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >`, 3765
- `std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >`, 3801
- `std::unordered_set< _Value, _Hash, _Pred, _Alloc >`, 3836
- skipws
  - `std`, 786
  - `std::basic_fstream< _CharT, _Traits >`, 1479
  - `std::basic_ifstream< _CharT, _Traits >`, 1535
  - `std::basic_ios< _CharT, _Traits >`, 1568
  - `std::basic_iostream< _CharT, _Traits >`, 1633
  - `std::basic_istream< _CharT, _Traits >`, 1685
  - `std::basic_istreamstream< _CharT, _Traits, _Alloc >`, 1739
  - `std::basic_ofstream< _CharT, _Traits >`, 1784
  - `std::basic_ostream< _CharT, _Traits >`, 1827
  - `std::basic_ostreamstream< _CharT, _Traits, _Alloc >`, 1872
  - `std::basic_stringstream< _CharT, _Traits, _Alloc >`, 2097
  - `std::ios_base`, 2723
- sleep\_for
  - `std::this_thread`, 857
- sleep\_until
  - `std::this_thread`, 857
- slice
  - Numeric Arrays, 266
- slice\_array
  - Numeric Arrays, 266
- `slice_array.h`, 4268
- slist, 4268
- snextc
  - `__gnu_cxx::enc_filebuf< _CharT >`, 2500
  - `__gnu_cxx::stdio_filebuf< _CharT, _Traits >`, 3515
  - `__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >`, 3539
  - `std::basic_filebuf< _CharT, _Traits >`, 1403
  - `std::basic_streambuf< _CharT, _Traits >`, 1903
  - `std::basic_stringbuf< _CharT, _Traits, _Alloc >`, 2024
  - `std::wbuffer_convert< _Codecvt, _Elem, _Tr >`, 3920
- sort
  - Sorting, 435, 436
  - `std::forward_list< _Tp, _Alloc >`, 2572
  - `std::list< _Tp, _Alloc >`, 2855
- `sort.h`, 4269
- `sort_heap`
  - Heap, 120, 121
- `sort_minimal_n`
  - `__gnu_parallel::Settings`, 1286
- `sort_mwms_oversampling`
  - `__gnu_parallel::Settings`, 1286
- `sort_qs_num_samples_preset`
  - `__gnu_parallel::Settings`, 1286
- `sort_qsb_base_case_maximal_n`
  - `__gnu_parallel::Settings`, 1286
- Sorting, 413
  - `inplace_merge`, 415, 416
  - `is_sorted`, 416, 417
  - `is_sorted_until`, 417, 418
  - `lexicographical_compare`, 418, 419
  - `max`, 419, 420
  - `max_element`, 421
  - `merge`, 422
  - `min`, 424, 425
  - `min_element`, 425, 426
  - `minmax`, 426, 427
  - `minmax_element`, 427, 428
  - `next_permutation`, 428, 429
  - `nth_element`, 429, 430
  - `partial_sort`, 430, 431
  - `partial_sort_copy`, 432
  - `prev_permutation`, 434
  - `sort`, 435, 436
  - `stable_sort`, 436, 437
- `span`, 4270
- `specfun.h`, 4270
- `sph_bessel`
  - Mathematical Special Functions, 179
  - TR1 Mathematical Special Functions, 447
- `sph_besself`
  - Mathematical Special Functions, 180
- `sph_bessell`
  - Mathematical Special Functions, 180
- `sph_legendre`
  - Mathematical Special Functions, 180
  - TR1 Mathematical Special Functions, 447
- `sph_legendref`
  - Mathematical Special Functions, 181
- `sph_legendrel`
  - Mathematical Special Functions, 181
- `sph_neumann`
  - Mathematical Special Functions, 182
  - TR1 Mathematical Special Functions, 447
- `sph_neumannf`
  - Mathematical Special Functions, 183
- `sph_neumannl`
  - Mathematical Special Functions, 183
- `splay_fnimps.hpp`, 4273
- `splay_tree.hpp`, 4273

- splice
  - std::list< \_Tp, \_Alloc >, 2855–2857
- splice\_after
  - std::forward\_list< \_Tp, \_Alloc >, 2573, 2574
- split\_fn\_imps.hpp, 4274
- split\_join\_can\_throw
  - \_\_gnu\_pbds::container\_traits< Cntr >, 2294
- split\_join\_fn\_imps.hpp, 4274, 4275
- sputbackc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2500
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3515
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3540
  - std::basic\_filebuf< \_CharT, \_Traits >, 1403
  - std::basic\_streambuf< \_CharT, \_Traits >, 1904
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2024
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3921
- sputc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2501
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3516
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3540
  - std::basic\_filebuf< \_CharT, \_Traits >, 1404
  - std::basic\_streambuf< \_CharT, \_Traits >, 1904
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2025
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3921
- sputn
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2501
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3516
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3541
  - std::basic\_filebuf< \_CharT, \_Traits >, 1404
  - std::basic\_streambuf< \_CharT, \_Traits >, 1905
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2025
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3922
- sqrt
  - Complex Numbers, 68
- sregex\_token\_iterator
  - Regular Expressions, 349
- sso\_string\_base.h, 4275
- sstream, 4275
- sstream.tcc, 4276
- ssub\_match
  - Regular Expressions, 350
- stable\_partition
  - Mutating, 216
- stable\_sort
  - Sorting, 436, 437
- stack, 4277
  - std::stack< \_Tp, \_Sequence >, 3494
- standard\_policies.hpp, 4277
- start
  - Numeric Arrays, 300, 301
- state
  - std::fpos< \_StateT >, 2578, 2579
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3922
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc >, 3937
- static\_pointer\_cast
  - Pointer Abstractions, 321
  - std, 787
- std, 587
  - \_\_Construct, 703
  - \_\_Destroy, 703, 704
  - \_\_Destroy\_n, 704
  - \_\_allocate\_guarded, 691
  - \_\_final\_insertion\_sort, 691
  - \_\_find\_if, 692
  - \_\_find\_if\_not, 692
  - \_\_find\_if\_not\_n, 693
  - \_\_gcd, 693
  - \_\_gen\_two\_uniform\_ints, 693
  - \_\_heap\_select, 694
  - \_\_inplace\_stable\_sort, 694
  - \_\_insertion\_sort, 694
  - \_\_introsort\_loop, 695
  - \_\_ioinit, 793
  - \_\_lg, 695
  - \_\_merge\_adaptive, 695
  - \_\_merge\_without\_buffer, 696
  - \_\_move\_median\_to\_first, 696
  - \_\_move\_merge, 696
  - \_\_move\_merge\_adaptive, 697
  - \_\_move\_merge\_adaptive\_backward, 697
  - \_\_partition, 697, 698
  - \_\_ptr\_rebind, 687
  - \_\_reverse, 698
  - \_\_rotate\_adaptive, 699
  - \_\_sample, 699
  - \_\_search\_n\_aux, 700
  - \_\_stable\_partition\_adaptive, 700
  - \_\_umap\_traits, 687
  - \_\_ummap\_traits, 687
  - \_\_umset\_traits, 687
  - \_\_unguarded\_insertion\_sort, 701
  - \_\_unguarded\_linear\_insert, 701
  - \_\_unguarded\_partition, 701
  - \_\_unguarded\_partition\_pivot, 702
  - \_\_unique\_copy, 702, 703
  - \_\_uset\_traits, 687
- acos, 704
- acosh, 705
- advance, 705
- arg, 706
- asin, 706
- asinh, 706
- atan, 706
- atanh, 706

begin, 707  
boolalpha, 709  
cbegin, 709  
cend, 709  
cerr, 793  
chars\_format, 690  
cin, 793  
clog, 794  
const\_pointer\_cast, 710  
cout, 794  
crbegin, 710  
crend, 710  
dec, 711  
defaultfloat, 711  
denorm\_absent, 691  
denorm\_indeterminate, 691  
denorm\_present, 691  
distance, 711  
dynamic\_pointer\_cast, 712  
end, 712, 713  
endl, 713  
ends, 713  
exchange, 714  
fabs, 714  
fixed, 714  
float\_denorm\_style, 690  
float\_round\_style, 691  
flush, 714  
from\_chars, 715  
get\_money, 715  
get\_new\_handler, 715  
get\_temporary\_buffer, 716  
get\_time, 716  
getline, 717–719  
hex, 719  
hexfloat, 720  
index\_sequence, 688  
index\_sequence\_for, 688  
internal, 720  
io\_errc, 691  
is\_nothrow\_swappable\_v, 794  
is\_nothrow\_swappable\_with\_v, 794  
is\_swappable\_v, 794  
is\_swappable\_with\_v, 795  
isalnum, 720  
isalpha, 720  
isblank, 721  
iscntrl, 721  
isdigit, 721  
isgraph, 721  
islower, 722  
isprint, 722  
ispunct, 722  
isspace, 722  
isupper, 723  
isxdigit, 723  
left, 723  
make\_index\_sequence, 688  
make\_integer\_sequence, 688  
new\_handler, 688  
noboolalpha, 723  
noshowbase, 724  
noshowpoint, 724  
noshowpos, 724  
noskipws, 724  
nounitbuf, 725  
nouppercase, 725  
oct, 725  
operator!=, 725–730  
operator<, 733–740  
operator<<, 740–747  
operator<=, 748–751  
operator>, 761–764  
operator>>, 768, 770–776  
operator>=, 765–768  
operator^, 777  
operator+, 731–733  
operator==, 752–755, 757–760  
operator&, 730  
operator|, 777  
put\_money, 778  
put\_time, 778  
quoted, 779  
rbegin, 779, 780  
rend, 780, 781  
replace\_copy, 782  
resetiosflags, 782  
return\_temporary\_buffer, 783  
right, 783  
round\_to\_nearest, 691  
round\_toward\_infinity, 691  
round\_toward\_neg\_infinity, 691  
round\_toward\_zero, 691  
scientific, 783  
set\_new\_handler, 784  
setbase, 784  
setfill, 784  
setiosflags, 785  
setprecision, 785  
setw, 785  
showbase, 786  
showpoint, 786  
showpos, 786  
skipws, 786  
static\_pointer\_cast, 787  
streamoff, 689  
streampos, 689  
streamsize, 689

- swap, [787–791](#)
- tolower, [792](#)
- toupper, [792](#)
- u16streampos, [689](#)
- u32streampos, [689](#)
- unitbuf, [792](#)
- uppercase, [792](#)
- wcerr, [795](#)
- wcin, [795](#)
- wclog, [795](#)
- wcout, [795](#)
- ws, [792](#)
- wstreampos, [690](#)
- std::\_\_add\_pointer\_helper< \_Tp, bool >, [864](#)
- std::\_\_allocated\_ptr< \_Alloc >, [878](#)
- \_\_allocated\_ptr, [878](#), [879](#)
- ~\_\_allocated\_ptr, [879](#)
- get, [879](#)
- operator=, [879](#)
- std::\_\_atomic\_base< \_ITp >, [880](#)
- std::\_\_atomic\_base< \_PTp \* >, [882](#)
- std::\_\_atomic\_flag\_base, [883](#)
- std::\_\_basic\_future< \_Res >, [884](#)
- \_M\_get\_result, [885](#)
- \_Ptr, [885](#)
- std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, [889](#)
- do\_out, [891](#)
- in, [891](#)
- out, [892](#)
- unshift, [893](#)
- std::\_\_ctype\_abstract\_base< \_CharT >, [899](#)
- char\_type, [901](#)
- do\_is, [901](#)
- do\_narrow, [902](#), [903](#)
- do\_scan\_is, [903](#)
- do\_scan\_not, [904](#)
- do\_tolower, [905](#)
- do\_toupper, [906](#)
- do\_widen, [907](#), [908](#)
- is, [908](#), [909](#)
- narrow, [909](#), [910](#)
- scan\_is, [911](#)
- scan\_not, [911](#)
- tolower, [912](#)
- toupper, [914](#)
- widen, [915](#)
- std::\_\_debug, [796](#)
- operator<=, [800](#)
- operator>, [800](#)
- operator>=, [801](#)
- swap, [801](#)
- std::\_\_debug::bitset< \_Nb >, [2166](#)
- std::\_\_debug::deque< \_Tp, \_Allocator >, [2436](#)
- \_M\_const\_iterators, [2440](#)
- \_M\_detach\_all, [2439](#)
- \_M\_detach\_singular, [2439](#)
- \_M\_get\_mutex, [2439](#)
- \_M\_invalidate\_all, [2439](#)
- \_M\_invalidate\_if, [2439](#)
- \_M\_iterators, [2440](#)
- \_M\_revalidate\_singular, [2440](#)
- \_M\_swap, [2440](#)
- \_M\_transfer\_from\_if, [2440](#)
- \_M\_version, [2441](#)
- std::\_\_debug::forward\_list< \_Tp, \_Alloc >, [2545](#)
- \_M\_const\_iterators, [2549](#)
- \_M\_detach\_all, [2548](#)
- \_M\_detach\_singular, [2548](#)
- \_M\_get\_mutex, [2548](#)
- \_M\_invalidate\_all, [2548](#)
- \_M\_invalidate\_if, [2549](#)
- \_M\_iterators, [2550](#)
- \_M\_revalidate\_singular, [2549](#)
- \_M\_transfer\_from\_if, [2549](#)
- \_M\_version, [2550](#)
- std::\_\_debug::list< \_Tp, \_Allocator >, [2825](#)
- \_M\_const\_iterators, [2830](#)
- \_M\_detach\_all, [2828](#)
- \_M\_detach\_singular, [2828](#)
- \_M\_get\_mutex, [2828](#)
- \_M\_invalidate\_all, [2828](#)
- \_M\_invalidate\_if, [2829](#)
- \_M\_iterators, [2830](#)
- \_M\_revalidate\_singular, [2829](#)
- \_M\_swap, [2829](#)
- \_M\_transfer\_from\_if, [2829](#)
- \_M\_version, [2830](#)
- std::\_\_debug::map< \_Key, \_Tp, \_Compare, \_Allocator >, [2896](#)
- \_M\_const\_iterators, [2901](#)
- \_M\_detach\_all, [2899](#)
- \_M\_detach\_singular, [2899](#)
- \_M\_get\_mutex, [2899](#)
- \_M\_invalidate\_all, [2899](#)
- \_M\_invalidate\_if, [2900](#)
- \_M\_iterators, [2901](#)
- \_M\_revalidate\_singular, [2900](#)
- \_M\_swap, [2900](#)
- \_M\_transfer\_from\_if, [2900](#)
- \_M\_version, [2901](#)
- std::\_\_debug::multimap< \_Key, \_Tp, \_Compare, \_Allocator >, [3015](#)
- \_M\_const\_iterators, [3020](#)
- \_M\_detach\_all, [3018](#)
- \_M\_detach\_singular, [3018](#)
- \_M\_get\_mutex, [3019](#)
- \_M\_invalidate\_all, [3019](#)

- [\\_M\\_invalidate\\_if, 3019](#)
- [\\_M\\_iterators, 3020](#)
- [\\_M\\_revalidate\\_singular, 3019](#)
- [\\_M\\_swap, 3020](#)
- [\\_M\\_transfer\\_from\\_if, 3020](#)
- [\\_M\\_version, 3020](#)
- [std::\\_\\_debug::multiset< \\_Key, \\_Compare, \\_Allocator >, 3080](#)
  - [\\_M\\_const\\_iterators, 3085](#)
  - [\\_M\\_detach\\_all, 3083](#)
  - [\\_M\\_detach\\_singular, 3083](#)
  - [\\_M\\_get\\_mutex, 3083](#)
  - [\\_M\\_invalidate\\_all, 3083](#)
  - [\\_M\\_invalidate\\_if, 3084](#)
  - [\\_M\\_iterators, 3085](#)
  - [\\_M\\_revalidate\\_singular, 3084](#)
  - [\\_M\\_swap, 3084](#)
  - [\\_M\\_transfer\\_from\\_if, 3084](#)
  - [\\_M\\_version, 3085](#)
- [std::\\_\\_debug::set< \\_Key, \\_Compare, \\_Allocator >, 3446](#)
  - [\\_M\\_const\\_iterators, 3451](#)
  - [\\_M\\_detach\\_all, 3449](#)
  - [\\_M\\_detach\\_singular, 3449](#)
  - [\\_M\\_get\\_mutex, 3449](#)
  - [\\_M\\_invalidate\\_all, 3449](#)
  - [\\_M\\_invalidate\\_if, 3450](#)
  - [\\_M\\_iterators, 3451](#)
  - [\\_M\\_revalidate\\_singular, 3450](#)
  - [\\_M\\_swap, 3450](#)
  - [\\_M\\_transfer\\_from\\_if, 3450](#)
  - [\\_M\\_version, 3451](#)
- [std::\\_\\_debug::unordered\\_map< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3753](#)
  - [\\_M\\_const\\_iterators, 3757](#)
  - [\\_M\\_const\\_local\\_iterators, 3757](#)
  - [\\_M\\_detach\\_all, 3755](#)
  - [\\_M\\_detach\\_singular, 3756](#)
  - [\\_M\\_get\\_mutex, 3756](#)
  - [\\_M\\_invalidate\\_all, 3756](#)
  - [\\_M\\_invalidate\\_if, 3756](#)
  - [\\_M\\_invalidate\\_local\\_if, 3756](#)
  - [\\_M\\_iterators, 3758](#)
  - [\\_M\\_local\\_iterators, 3758](#)
  - [\\_M\\_revalidate\\_singular, 3757](#)
  - [\\_M\\_swap, 3757](#)
  - [\\_M\\_version, 3758](#)
- [std::\\_\\_debug::unordered\\_multimap< \\_Key, \\_Tp, \\_Hash, \\_Pred, \\_Alloc >, 3788](#)
  - [\\_M\\_const\\_iterators, 3793](#)
  - [\\_M\\_const\\_local\\_iterators, 3793](#)
  - [\\_M\\_detach\\_all, 3791](#)
  - [\\_M\\_detach\\_singular, 3791](#)
  - [\\_M\\_get\\_mutex, 3792](#)
  - [\\_M\\_invalidate\\_all, 3792](#)
- [\\_M\\_invalidate\\_if, 3792](#)
- [\\_M\\_invalidate\\_local\\_if, 3792](#)
- [\\_M\\_iterators, 3794](#)
- [\\_M\\_local\\_iterators, 3794](#)
- [\\_M\\_revalidate\\_singular, 3793](#)
- [\\_M\\_swap, 3793](#)
- [\\_M\\_version, 3794](#)
- [std::\\_\\_debug::unordered\\_multiset< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3824](#)
  - [\\_M\\_const\\_iterators, 3829](#)
  - [\\_M\\_const\\_local\\_iterators, 3829](#)
  - [\\_M\\_detach\\_all, 3827](#)
  - [\\_M\\_detach\\_singular, 3827](#)
  - [\\_M\\_get\\_mutex, 3827](#)
  - [\\_M\\_invalidate\\_all, 3827](#)
  - [\\_M\\_invalidate\\_if, 3828](#)
  - [\\_M\\_invalidate\\_local\\_if, 3828](#)
  - [\\_M\\_iterators, 3829](#)
  - [\\_M\\_local\\_iterators, 3829](#)
  - [\\_M\\_revalidate\\_singular, 3828](#)
  - [\\_M\\_swap, 3828](#)
  - [\\_M\\_version, 3830](#)
- [std::\\_\\_debug::unordered\\_set< \\_Value, \\_Hash, \\_Pred, \\_Alloc >, 3859](#)
  - [\\_M\\_const\\_iterators, 3864](#)
  - [\\_M\\_const\\_local\\_iterators, 3864](#)
  - [\\_M\\_detach\\_all, 3862](#)
  - [\\_M\\_detach\\_singular, 3862](#)
  - [\\_M\\_get\\_mutex, 3862](#)
  - [\\_M\\_invalidate\\_all, 3862](#)
  - [\\_M\\_invalidate\\_if, 3863](#)
  - [\\_M\\_invalidate\\_local\\_if, 3863](#)
  - [\\_M\\_iterators, 3864](#)
  - [\\_M\\_local\\_iterators, 3864](#)
  - [\\_M\\_revalidate\\_singular, 3863](#)
  - [\\_M\\_swap, 3863](#)
  - [\\_M\\_version, 3865](#)
- [std::\\_\\_debug::vector< \\_Tp, \\_Allocator >, 3897](#)
  - [\\_M\\_const\\_iterators, 3901](#)
  - [\\_M\\_detach\\_all, 3900](#)
  - [\\_M\\_detach\\_singular, 3900](#)
  - [\\_M\\_get\\_mutex, 3900](#)
  - [\\_M\\_invalidate\\_all, 3900](#)
  - [\\_M\\_invalidate\\_if, 3900](#)
  - [\\_M\\_iterators, 3901](#)
  - [\\_M\\_revalidate\\_singular, 3901](#)
  - [\\_M\\_swap, 3901](#)
  - [\\_M\\_transfer\\_from\\_if, 3901](#)
  - [\\_M\\_version, 3902](#)
- [vector, 3899](#)
- [std::\\_\\_detail, 801](#)
  - [\\_\\_from\\_chars\\_alnum, 805](#)
  - [\\_\\_from\\_chars\\_binary, 805](#)
  - [\\_\\_from\\_chars\\_digit, 805](#)

- operator<<, [806](#)
- operator>>, [806](#)
- std::\_\_detail::\_\_BracketMatcher< \_TraitsT, \_\_icase, \_\_collate >, [1051](#)
- std::\_\_detail::\_\_Compiler< \_TraitsT >, [1054](#)
- std::\_\_detail::\_\_Default\_ranged\_hash, [1055](#)
- std::\_\_detail::\_\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >, [1070](#)
- std::\_\_detail::\_\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >, [1070](#)
- std::\_\_detail::\_\_Equality< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >, [1071](#)
- std::\_\_detail::\_\_Executor< \_Bilter, \_Alloc, \_TraitsT, \_\_dfs\_mode >, [1073](#)
- std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, false >, [1088](#)
- std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Default\_ranged\_hash, true >, [1090](#)
- std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >, [1088](#)
- std::\_\_detail::\_\_Hash\_code\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, false >, [1091](#)
- std::\_\_detail::\_\_Hash\_node< \_Value, \_Cache\_hash\_code >, [1092](#)
- std::\_\_detail::\_\_Hash\_node< \_Value, false >, [1093](#)
- std::\_\_detail::\_\_Hash\_node< \_Value, true >, [1094](#)
- std::\_\_detail::\_\_Hash\_node\_base, [1095](#)
- std::\_\_detail::\_\_Hash\_node\_value\_base< \_Value >, [1096](#)
- std::\_\_detail::\_\_Hashtable\_alloc< \_NodeAlloc >, [1103](#)
- std::\_\_detail::\_\_Hashtable\_base< \_Key, \_Value, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_Traits >, [1104](#)
- std::\_\_detail::\_\_Hashtable\_ebo\_helper< \_Nm, \_Tp, \_\_use\_ebo >, [1106](#)
- std::\_\_detail::\_\_Hashtable\_ebo\_helper< \_Nm, \_Tp, false >, [1106](#)
- std::\_\_detail::\_\_Hashtable\_ebo\_helper< \_Nm, \_Tp, true >, [1106](#)
- std::\_\_detail::\_\_Hashtable\_traits< \_\_Cache\_hash\_code, \_\_Constant\_iterators, \_\_Unique\_keys >, [1107](#)
- std::\_\_detail::\_\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_\_Constant\_iterators >, [1111](#)
- std::\_\_detail::\_\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >, [1111](#)
- std::\_\_detail::\_\_Insert< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >, [1113](#)
- std::\_\_detail::\_\_Insert\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >, [1115](#)
- std::\_\_detail::\_\_List\_node\_base, [1137](#)
- std::\_\_detail::\_\_List\_node\_header, [1138](#)
- std::\_\_detail::\_\_Local\_const\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >, [1139](#)
- std::\_\_detail::\_\_Local\_iterator< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_constant\_iterators, \_\_cache >, [1140](#)
- std::\_\_detail::\_\_Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, \_\_cache\_hash\_code >, [1141](#)
- std::\_\_detail::\_\_Local\_iterator\_base< \_Key, \_Value, \_ExtractKey, \_H1, \_H2, \_Hash, true >, [1142](#)
- std::\_\_detail::\_\_Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false >, [1165](#)
- std::\_\_detail::\_\_Map\_base< \_Key, \_Pair, \_Alloc, \_Select1st, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true >, [1165](#)
- std::\_\_detail::\_\_Map\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, \_Unique\_keys >, [1164](#)
- std::\_\_detail::\_\_Mask\_range\_hashing, [1166](#)
- std::\_\_detail::\_\_Mod\_range\_hashing, [1168](#)
- std::\_\_detail::\_\_Node\_const\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >, [1177](#)
- std::\_\_detail::\_\_Node\_iterator< \_Value, \_\_constant\_iterators, \_\_cache >, [1182](#)
- std::\_\_detail::\_\_Node\_iterator\_base< \_Value, \_Cache\_hash\_code >, [1183](#)
- std::\_\_detail::\_\_Power2\_rehash\_policy, [1194](#)
- std::\_\_detail::\_\_Prime\_rehash\_policy, [1195](#)
- std::\_\_detail::\_\_Quoted\_string< \_String, \_CharT >, [1201](#)
- std::\_\_detail::\_\_Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, false\_type >, [1204](#)
- std::\_\_detail::\_\_Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, true\_type >, [1204](#)
- std::\_\_detail::\_\_Rehash\_base< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits, typename >, [1203](#)
- std::\_\_detail::\_\_Scanner< \_CharT >, [1274](#)
- std::\_\_detail::\_\_TokenT, [1276](#)
- std::\_\_detail::\_\_StateSeq< \_TraitsT >, [1290](#)
- std::\_\_detector< \_Default, \_\_void\_t< \_Op< \_Args... > >, \_Op, \_Args... >, [917](#)
- std::\_\_detector< \_Default, \_AlwaysVoid, \_Op, \_Args >, [916](#)
- std::\_\_exception\_ptr::exception\_ptr, [2523](#)
- operator==, [2524](#)



std::\_\_future\_base, 927  
     Ptr, 928  
 std::\_\_future\_base::Result< \_Res >, 1208  
 std::\_\_future\_base::Result< \_Res & >, 1209  
 std::\_\_future\_base::Result< void >, 1210  
 std::\_\_future\_base::Result\_alloc< \_Res, \_Alloc >, 1211  
 std::\_\_future\_base::Result\_base, 1212  
 std::\_\_is\_location\_invariant< \_Tp >, 937  
 std::\_\_is\_nullptr\_t< \_Tp >, 937  
 std::\_\_is\_tuple\_like\_impl< std::pair< \_T1, \_T2 > >, 938  
 std::\_\_numeric\_limits\_base, 948  
     digits, 949  
     digits10, 950  
     has\_denorm, 950  
     has\_denorm\_loss, 950  
     has\_infinity, 950  
     has\_quiet\_NaN, 950  
     has\_signaling\_NaN, 951  
     is\_bounded, 951  
     is\_exact, 951  
     is\_iec559, 951  
     is\_integer, 951  
     is\_modulo, 952  
     is\_signed, 952  
     is\_specialized, 952  
     max\_digits10, 952  
     max\_exponent, 952  
     max\_exponent10, 953  
     min\_exponent, 953  
     min\_exponent10, 953  
     radix, 953  
     round\_style, 953  
     tinyness\_before, 954  
     traps, 954  
 std::\_\_parallel, 807  
 std::\_\_parallel::CRandNumber< \_MustBeInt >, 1055  
 std::\_\_Base\_bitset< 0 >, 1047  
 std::\_\_Base\_bitset< 1 >, 1048  
 std::\_\_Base\_bitset< \_Nw >, 1045  
     \_M\_w, 1047  
 std::\_\_Bind< \_Signature >, 1050  
 std::\_\_Bind\_result< \_Result, \_Signature >, 1050  
 std::\_\_Deque\_base< \_Tp, \_Alloc >, 1056  
     \_M\_initialize\_map, 1057  
 std::\_\_Deque\_iterator< \_Tp, \_Ref, \_Ptr >, 1058  
     \_M\_set\_node, 1059  
 std::\_\_Enable\_copy\_move< \_Copy, \_CopyAssignment, \_Move, \_MoveAssignment, \_Tag >, 1065  
 std::\_\_Enable\_default\_constructor< \_Switch, \_Tag >, 1066  
 std::\_\_Enable\_destructor< \_Switch, \_Tag >, 1066  
 std::\_\_Enable\_special\_members< \_Default, \_Destructor, \_Copy, \_CopyAssignment, \_Move, \_MoveAssignment, \_Tag >, 1067  
 std::\_\_Function\_base, 1077  
 std::\_\_Fwd\_list\_base< \_Tp, \_Alloc >, 1078  
 std::\_\_Fwd\_list\_const\_iterator< \_Tp >, 1080  
     operator!=, 1080  
     operator==, 1081  
 std::\_\_Fwd\_list\_iterator< \_Tp >, 1081  
     operator!=, 1082  
     operator==, 1082  
 std::\_\_Fwd\_list\_node< \_Tp >, 1083  
 std::\_\_Fwd\_list\_node\_base, 1084  
 std::\_\_Hashtable< \_Key, \_Value, \_Alloc, \_ExtractKey, \_Equal, \_H1, \_H2, \_Hash, \_RehashPolicy, \_Traits >, 1097  
 std::\_\_List\_base< \_Tp, \_Alloc >, 1132  
 std::\_\_List\_const\_iterator< \_Tp >, 1134  
 std::\_\_List\_iterator< \_Tp >, 1135  
 std::\_\_List\_node< \_Tp >, 1136  
 std::\_\_Mu< \_Arg, \_IsBindExp, \_IsPlaceholder >, 1168  
 std::\_\_Mu< \_Arg, false, false >, 1169  
 std::\_\_Mu< \_Arg, false, true >, 1169  
 std::\_\_Mu< \_Arg, true, false >, 1170  
 std::\_\_Mu< reference\_wrapper< \_Tp >, false, false >, 1170  
 std::\_\_Not\_fn< \_Fn >, 1184  
 std::\_\_Placeholder< \_Num >, 1187  
 std::\_\_Sp\_ebo\_helper< \_Nm, \_Tp, false >, 1288  
 std::\_\_Sp\_ebo\_helper< \_Nm, \_Tp, true >, 1288  
 std::\_\_Temporary\_buffer< \_ForwardIterator, \_Tp >, 1291  
     \_Temporary\_buffer, 1292  
     begin, 1293  
     end, 1293  
     requested\_size, 1293  
     size, 1293  
 std::\_\_Tuple\_impl< \_Idx, \_Elements >, 1294  
 std::\_\_Tuple\_impl< \_Idx, \_Head, \_Tail... >, 1294  
 std::\_\_V2::condition\_variable\_any, 2266  
 std::\_\_V2::error\_category, 2517  
 std::\_\_Vector\_base< \_Tp, \_Alloc >, 1296  
 std::add\_const< \_Tp >, 1297  
 std::add\_cv< \_Tp >, 1298  
 std::add\_lvalue\_reference< \_Tp >, 1298  
 std::add\_rvalue\_reference< \_Tp >, 1299  
 std::add\_volatile< \_Tp >, 1299  
 std::adopt\_lock\_t, 1300  
 std::aligned\_storage< \_Len, \_Align >, 1300  
 std::aligned\_union< \_Len, \_Types >, 1301  
     type, 1301  
 std::alignment\_of< \_Tp >, 1302  
 std::allocator< \_Tp >, 1303  
 std::allocator< void >, 1304  
 std::allocator\_arg\_t, 1305  
 std::allocator\_traits< \_Alloc >, 1305  
     allocate, 1309, 1310  
     allocator\_type, 1306  
     const\_pointer, 1307



- const\_void\_pointer, 1307
- construct, 1310
- deallocate, 1311
- destroy, 1311
- difference\_type, 1307
- is\_always\_equal, 1307
- max\_size, 1312
- pointer, 1308
- propagate\_on\_container\_copy\_assignment, 1308
- propagate\_on\_container\_move\_assignment, 1308
- propagate\_on\_container\_swap, 1308
- select\_on\_container\_copy\_construction, 1312
- size\_type, 1309
- value\_type, 1309
- void\_pointer, 1309
- std::allocator\_traits< allocator< \_Tp > >, 1313
  - allocate, 1316, 1317
  - allocator\_type, 1314
  - const\_pointer, 1314
  - const\_void\_pointer, 1314
  - construct, 1317
  - deallocate, 1318
  - destroy, 1318
  - difference\_type, 1314
  - is\_always\_equal, 1315
  - max\_size, 1319
  - pointer, 1315
  - propagate\_on\_container\_copy\_assignment, 1315
  - propagate\_on\_container\_move\_assignment, 1315
  - propagate\_on\_container\_swap, 1315
  - select\_on\_container\_copy\_construction, 1319
  - size\_type, 1316
  - value\_type, 1316
  - void\_pointer, 1316
- std::array< \_Tp, \_Nm >, 1326
- std::atomic< \_Tp >, 1328
- std::atomic< \_Tp \* >, 1329
- std::atomic< bool >, 1331
- std::atomic< char >, 1332
- std::atomic< char16\_t >, 1334
- std::atomic< char32\_t >, 1336
- std::atomic< int >, 1338
- std::atomic< long >, 1340
- std::atomic< long long >, 1342
- std::atomic< short >, 1344
- std::atomic< signed char >, 1346
- std::atomic< unsigned char >, 1348
- std::atomic< unsigned int >, 1350
- std::atomic< unsigned long >, 1352
- std::atomic< unsigned long long >, 1354
- std::atomic< unsigned short >, 1356
- std::atomic< wchar\_t >, 1358
- std::atomic\_flag, 1360
- std::auto\_ptr< \_Tp >, 1361
  - ~auto\_ptr, 1364
  - auto\_ptr, 1362–1364
  - element\_type, 1362
  - get, 1364
  - operator\*, 1364
  - operator->, 1365
  - operator=, 1365
  - release, 1366
  - reset, 1366
- std::auto\_ptr\_ref< \_Tp1 >, 1367
- std::back\_insert\_iterator< \_Container >, 1368
  - back\_insert\_iterator, 1370
  - container\_type, 1369
  - difference\_type, 1369
  - iterator\_category, 1369
  - operator\*, 1370
  - operator++, 1371
  - operator=, 1371
  - pointer, 1369
  - reference, 1370
  - value\_type, 1370
- std::bad\_alloc, 1372
  - what, 1372
- std::bad\_cast, 1374
  - what, 1375
- std::bad\_exception, 1375
  - what, 1376
- std::bad\_function\_call, 1376
  - what, 1377
- std::bad\_typeid, 1378
  - what, 1379
- std::bad\_weak\_ptr, 1379
  - what, 1380
- std::basic\_filebuf< \_CharT, \_Traits >, 1385
  - \_M\_buf, 1408
  - \_M\_buf\_locale, 1408
  - \_M\_buf\_size, 1408
  - \_M\_create\_pback, 1389
  - \_M\_destroy\_pback, 1390
  - \_M\_ext\_buf, 1408
  - \_M\_ext\_buf\_size, 1408
  - \_M\_ext\_next, 1409
  - \_M\_in\_beg, 1409
  - \_M\_in\_cur, 1409
  - \_M\_in\_end, 1409
  - \_M\_mode, 1409
  - \_M\_out\_beg, 1410
  - \_M\_out\_cur, 1410
  - \_M\_out\_end, 1410
  - \_M\_pback, 1410
  - \_M\_pback\_cur\_save, 1411
  - \_M\_pback\_end\_save, 1411
  - \_M\_pback\_init, 1411
  - \_M\_reading, 1412

- [\\_M\\_set\\_buffer, 1390](#)
- [~basic\\_filebuf, 1389](#)
- [basic\\_filebuf, 1389](#)
- [close, 1390](#)
- [eback, 1390](#)
- [egptr, 1391](#)
- [epptr, 1391](#)
- [gbump, 1391](#)
- [getloc, 1392](#)
- [gptr, 1392](#)
- [imbue, 1392](#)
- [in\\_avail, 1393](#)
- [is\\_open, 1393](#)
- [open, 1393, 1394](#)
- [overflow, 1395](#)
- [pbackfail, 1395](#)
- [pbase, 1396](#)
- [pbump, 1396](#)
- [pptr, 1397](#)
- [pubimbue, 1397](#)
- [pubseekoff, 1397](#)
- [pubseekpos, 1398](#)
- [pubsetbuf, 1398](#)
- [pubsync, 1399](#)
- [sbumpc, 1399](#)
- [seekoff, 1399](#)
- [seekpos, 1399](#)
- [setbuf, 1400](#)
- [setg, 1400](#)
- [setp, 1401](#)
- [sgetc, 1401](#)
- [sgetn, 1402](#)
- [showmanyc, 1402](#)
- [snextc, 1403](#)
- [sputbackc, 1403](#)
- [sputc, 1404](#)
- [sputn, 1404](#)
- [sungetc, 1405](#)
- [sync, 1405](#)
- [uflow, 1405](#)
- [underflow, 1406](#)
- [xsgetn, 1406](#)
- [xsputn, 1407](#)
- [std::basic\\_fstream<\\_CharT, \\_Traits >, 1413](#)
  - [\\_M\\_gcount, 1472](#)
  - [\\_M\\_getloc, 1424](#)
  - [\\_M\\_write, 1424](#)
  - [\\_\\_num\\_put\\_type, 1420](#)
  - [~basic\\_fstream, 1424](#)
  - [adjustfield, 1472](#)
  - [app, 1472](#)
  - [ate, 1472](#)
  - [bad, 1425](#)
  - [badbit, 1472](#)
  - [basefield, 1473](#)
  - [basic\\_fstream, 1423](#)
  - [beg, 1473](#)
  - [binary, 1473](#)
  - [boolalpha, 1473](#)
  - [clear, 1425](#)
  - [close, 1425](#)
  - [copyfmt, 1426](#)
  - [cur, 1474](#)
  - [dec, 1474](#)
  - [end, 1474](#)
  - [eof, 1426](#)
  - [eofbit, 1474](#)
  - [event, 1422](#)
  - [event\\_callback, 1420](#)
  - [exceptions, 1426, 1427](#)
  - [fail, 1427](#)
  - [failbit, 1475](#)
  - [fill, 1428](#)
  - [fixed, 1475](#)
  - [flags, 1429](#)
  - [floatfield, 1475](#)
  - [flush, 1429](#)
  - [fmtflags, 1420](#)
  - [gcount, 1430](#)
  - [get, 1430–1433](#)
  - [getline, 1433–1435](#)
  - [getloc, 1435](#)
  - [good, 1435](#)
  - [goodbit, 1475](#)
  - [hex, 1476](#)
  - [ignore, 1435, 1436](#)
  - [imbue, 1437](#)
  - [in, 1476](#)
  - [init, 1437](#)
  - [internal, 1476](#)
  - [iostate, 1421](#)
  - [is\\_open, 1438](#)
  - [iword, 1438](#)
  - [left, 1477](#)
  - [narrow, 1438](#)
  - [oct, 1477](#)
  - [open, 1439](#)
  - [openmode, 1421](#)
  - [operator bool, 1440](#)
  - [operator!, 1440](#)
  - [operator<<, 1440–1448](#)
  - [operator>>, 1448–1456](#)
  - [out, 1477](#)
  - [peek, 1456](#)
  - [precision, 1457](#)
  - [put, 1458](#)
  - [putback, 1458](#)
  - [pword, 1459](#)

- rdbuf, 1459, 1460
- rdstate, 1460
- read, 1460
- readsome, 1461
- register\_callback, 1462
- right, 1477
- scientific, 1478
- seekdir, 1422
- seekg, 1462, 1463
- seekp, 1463, 1464
- setf, 1464, 1465
- setstate, 1465
- showbase, 1478
- showpoint, 1478
- showpos, 1478
- skipws, 1479
- sync, 1466
- sync\_with\_stdio, 1466
- tellg, 1467
- tellp, 1467
- tie, 1467, 1468
- trunc, 1479
- unget, 1468
- unitbuf, 1479
- unsetf, 1469
- uppercase, 1479
- widen, 1469
- width, 1470
- write, 1471
- xalloc, 1471
- std::basic\_ifstream< \_CharT, \_Traits >, 1482
  - \_M\_gcount, 1528
  - \_M\_getloc, 1492
  - \_\_num\_put\_type, 1488
  - ~basic\_ifstream, 1492
  - adjustfield, 1528
  - app, 1528
  - ate, 1529
  - bad, 1492
  - badbit, 1529
  - basefield, 1529
  - basic\_ifstream, 1491
  - beg, 1529
  - binary, 1529
  - boolalpha, 1530
  - clear, 1493
  - close, 1493
  - copyfmt, 1493
  - cur, 1530
  - dec, 1530
  - end, 1530
  - eof, 1494
  - eofbit, 1531
  - event, 1490
  - event\_callback, 1488
  - exceptions, 1494
  - fail, 1495
  - failbit, 1531
  - fill, 1495, 1496
  - fixed, 1531
  - flags, 1496, 1497
  - floatfield, 1531
  - fmtflags, 1488
  - gcount, 1497
  - get, 1497–1500
  - getline, 1501, 1502
  - getloc, 1502
  - good, 1502
  - goodbit, 1532
  - hex, 1532
  - ignore, 1503, 1504
  - imbue, 1504
  - in, 1532
  - init, 1505
  - internal, 1532
  - iostate, 1489
  - is\_open, 1505
  - isword, 1505
  - left, 1533
  - narrow, 1506
  - oct, 1533
  - open, 1506, 1507
  - openmode, 1489
  - operator bool, 1507
  - operator!, 1507
  - operator>>, 1508–1515
  - out, 1533
  - peek, 1515
  - precision, 1516
  - putback, 1517
  - pword, 1517
  - rdbuf, 1518
  - rdstate, 1518
  - read, 1519
  - readsome, 1519
  - register\_callback, 1520
  - right, 1533
  - scientific, 1534
  - seekdir, 1490
  - seekg, 1521
  - setf, 1522
  - setstate, 1523
  - showbase, 1534
  - showpoint, 1534
  - showpos, 1534
  - skipws, 1535
  - sync, 1523
  - sync\_with\_stdio, 1523

- tellg, [1524](#)
- tie, [1524](#), [1525](#)
- trunc, [1535](#)
- unget, [1525](#)
- unitbuf, [1535](#)
- unsetf, [1526](#)
- uppercase, [1535](#)
- widen, [1526](#)
- width, [1527](#)
- xalloc, [1528](#)
- std::basic\_ios<\_CharT, \_Traits >, [1537](#)
  - \_M\_getloc, [1546](#)
  - \_\_ctype\_type, [1540](#)
  - \_\_num\_get\_type, [1540](#)
  - \_\_num\_put\_type, [1541](#)
  - ~basic\_ios, [1545](#)
  - adjustfield, [1561](#)
  - app, [1561](#)
  - ate, [1562](#)
  - bad, [1546](#)
  - badbit, [1562](#)
  - basefield, [1562](#)
  - basic\_ios, [1545](#), [1546](#)
  - beg, [1562](#)
  - binary, [1562](#)
  - boolalpha, [1563](#)
  - char\_type, [1541](#)
  - clear, [1547](#)
  - copyfmt, [1547](#)
  - cur, [1563](#)
  - dec, [1563](#)
  - end, [1563](#)
  - eof, [1548](#)
  - eofbit, [1564](#)
  - event, [1545](#)
  - event\_callback, [1541](#)
  - exceptions, [1548](#)
  - fail, [1549](#)
  - failbit, [1564](#)
  - fill, [1549](#)
  - fixed, [1564](#)
  - flags, [1550](#)
  - floatfield, [1564](#)
  - fmtflags, [1542](#)
  - getloc, [1551](#)
  - good, [1551](#)
  - goodbit, [1565](#)
  - hex, [1565](#)
  - imbue, [1551](#)
  - in, [1565](#)
  - init, [1552](#)
  - int\_type, [1542](#)
  - internal, [1565](#)
  - iostate, [1543](#)
  - iword, [1552](#)
  - left, [1566](#)
  - narrow, [1553](#)
  - oct, [1566](#)
  - off\_type, [1543](#)
  - openmode, [1543](#)
  - operator bool, [1553](#)
  - operator!, [1553](#)
  - out, [1566](#)
  - pos\_type, [1544](#)
  - precision, [1554](#)
  - pword, [1554](#)
  - rdbuf, [1555](#)
  - rdstate, [1556](#)
  - register\_callback, [1556](#)
  - right, [1566](#)
  - scientific, [1567](#)
  - seekdir, [1544](#)
  - setf, [1557](#)
  - setstate, [1558](#)
  - showbase, [1567](#)
  - showpoint, [1567](#)
  - showpos, [1567](#)
  - skipws, [1568](#)
  - sync\_with\_stdio, [1558](#)
  - tie, [1558](#), [1559](#)
  - traits\_type, [1544](#)
  - trunc, [1568](#)
  - unitbuf, [1568](#)
  - unsetf, [1559](#)
  - uppercase, [1568](#)
  - widen, [1560](#)
  - width, [1560](#)
  - xalloc, [1561](#)
- std::basic\_iostream<\_CharT, \_Traits >, [1569](#)
  - \_M\_gcount, [1626](#)
  - \_M\_getloc, [1579](#)
  - \_M\_write, [1579](#)
  - \_\_num\_put\_type, [1576](#)
  - ~basic\_iostream, [1579](#)
  - adjustfield, [1626](#)
  - app, [1626](#)
  - ate, [1626](#)
  - bad, [1580](#)
  - badbit, [1626](#)
  - basefield, [1627](#)
  - basic\_iostream, [1579](#)
  - beg, [1627](#)
  - binary, [1627](#)
  - boolalpha, [1627](#)
  - clear, [1580](#)
  - copyfmt, [1580](#)
  - cur, [1628](#)
  - dec, [1628](#)

end, 1628  
eof, 1581  
eofbit, 1628  
event, 1578  
event\_callback, 1576  
exceptions, 1581  
fail, 1582  
failbit, 1629  
fill, 1582, 1583  
fixed, 1629  
flags, 1583, 1584  
floatfield, 1629  
flush, 1584  
fmtflags, 1576  
gcount, 1584  
get, 1585–1588  
getline, 1588, 1589  
getloc, 1590  
good, 1590  
goodbit, 1629  
hex, 1630  
ignore, 1590, 1591  
imbue, 1592  
in, 1630  
init, 1592  
internal, 1630  
iostate, 1577  
iword, 1593  
left, 1631  
narrow, 1593  
oct, 1631  
openmode, 1577  
operator bool, 1594  
operator!, 1594  
operator<<, 1594–1596, 1598–1602  
operator>>, 1603–1610  
out, 1631  
peek, 1611  
precision, 1611  
put, 1612  
putback, 1612  
pword, 1613  
rdbuf, 1613, 1614  
rdstate, 1614  
read, 1615  
readsome, 1615  
register\_callback, 1616  
right, 1631  
scientific, 1632  
seekdir, 1578  
seekg, 1617  
seekp, 1618  
setf, 1619  
setstate, 1620

showbase, 1632  
showpoint, 1632  
showpos, 1632  
skipws, 1633  
sync, 1620  
sync\_with\_stdio, 1620  
tellg, 1621  
tellp, 1621  
tie, 1621, 1622  
trunc, 1633  
unget, 1622  
unitbuf, 1633  
unsetf, 1623  
uppercase, 1633  
widen, 1623  
width, 1624  
write, 1625  
xalloc, 1625  
std::basic\_istream<\_CharT, \_Traits >, 1634  
    \_M\_gcount, 1678  
    \_M\_getloc, 1643  
    \_\_num\_put\_type, 1640  
    ~basic\_istream, 1643  
adjustfield, 1678  
app, 1679  
ate, 1679  
bad, 1643  
badbit, 1679  
basefield, 1679  
basic\_istream, 1643  
beg, 1679  
binary, 1680  
boolalpha, 1680  
clear, 1644  
copyfmt, 1644  
cur, 1680  
dec, 1680  
end, 1681  
eof, 1645  
eofbit, 1681  
event, 1642  
event\_callback, 1640  
exceptions, 1645  
fail, 1646  
failbit, 1681  
fill, 1646  
fixed, 1681  
flags, 1647  
floatfield, 1682  
fmtflags, 1640  
gcount, 1648  
get, 1648–1651  
getline, 1651–1653  
getloc, 1653

- good, 1653
- goodbit, 1682
- hex, 1682
- ignore, 1653, 1654
- imbue, 1655
- in, 1682
- init, 1655
- internal, 1683
- iostate, 1641
- isword, 1656
- left, 1683
- narrow, 1656
- oct, 1683
- openmode, 1641
- operator bool, 1657
- operator!, 1657
- operator>>, 1657–1659, 1661–1665
- out, 1683
- peek, 1666
- precision, 1666
- putback, 1667
- pword, 1667
- rdbuf, 1668
- rdstate, 1669
- read, 1669
- readsome, 1670
- register\_callback, 1671
- right, 1684
- scientific, 1684
- seekdir, 1642
- seekg, 1671, 1672
- setf, 1672, 1673
- setstate, 1673
- showbase, 1684
- showpoint, 1684
- showpos, 1685
- skipws, 1685
- sync, 1674
- sync\_with\_stdio, 1674
- tellg, 1675
- tie, 1675
- trunc, 1685
- unget, 1676
- unitbuf, 1685
- unsetf, 1676
- uppercase, 1686
- widen, 1677
- width, 1677
- xalloc, 1678
- std::basic\_istream<\_CharT, \_Traits>::sentry, 3411
  - operator bool, 3413
  - sentry, 3412
  - traits\_type, 3412
- std::basic\_istream<\_CharT, \_Traits, \_Alloc>, 1686
  - \_M\_gcount, 1732
  - \_M\_getloc, 1696
  - \_\_num\_put\_type, 1692
  - ~basic\_istream, 1696
  - adjustfield, 1732
  - app, 1733
  - ate, 1733
  - bad, 1696
  - badbit, 1733
  - basefield, 1733
  - basic\_istream, 1695
  - beg, 1733
  - binary, 1734
  - boolalpha, 1734
  - clear, 1697
  - copyfmt, 1697
  - cur, 1734
  - dec, 1734
  - end, 1735
  - eof, 1698
  - eofbit, 1735
  - event, 1694
  - event\_callback, 1692
  - exceptions, 1698
  - fail, 1699
  - failbit, 1735
  - fill, 1699
  - fixed, 1735
  - flags, 1700
  - floatfield, 1736
  - fmtflags, 1692
  - gcount, 1701
  - get, 1701–1704
  - getline, 1704–1706
  - getloc, 1706
  - good, 1706
  - goodbit, 1736
  - hex, 1736
  - ignore, 1706, 1707
  - imbue, 1708
  - in, 1736
  - init, 1708
  - internal, 1737
  - iostate, 1693
  - isword, 1709
  - left, 1737
  - narrow, 1709
  - oct, 1737
  - openmode, 1693
  - operator bool, 1710
  - operator!, 1710
  - operator>>, 1710–1712, 1714–1718
  - out, 1737
  - peek, 1719

- precision, 1719
- putback, 1720
- pword, 1720
- rdbuf, 1721
- rdstate, 1722
- read, 1722
- readsome, 1723
- register\_callback, 1724
- right, 1738
- scientific, 1738
- seekdir, 1694
- seekg, 1724, 1725
- setf, 1725, 1726
- setstate, 1726
- showbase, 1738
- showpoint, 1738
- showpos, 1739
- skipws, 1739
- str, 1727
- sync, 1727
- sync\_with\_stdio, 1728
- tellg, 1728
- tie, 1729
- trunc, 1739
- unget, 1730
- unitbuf, 1739
- unsetf, 1730
- uppercase, 1740
- widen, 1731
- width, 1731
- xalloc, 1732
- std::basic\_ofstream< \_CharT, \_Traits >, 1740
  - \_M\_getloc, 1749
  - \_M\_write, 1750
  - \_\_num\_get\_type, 1745
  - ~basic\_ofstream, 1749
  - adjustfield, 1777
  - app, 1778
  - ate, 1778
  - bad, 1750
  - badbit, 1778
  - basefield, 1778
  - basic\_ofstream, 1748, 1749
  - beg, 1778
  - binary, 1779
  - boolalpha, 1779
  - clear, 1751
  - close, 1751
  - copyfmt, 1751
  - cur, 1779
  - dec, 1779
  - end, 1780
  - eof, 1752
  - eofbit, 1780
  - event, 1748
  - event\_callback, 1746
  - exceptions, 1752
  - fail, 1753
  - failbit, 1780
  - fill, 1753
  - fixed, 1780
  - flags, 1754
  - floatfield, 1781
  - flush, 1755
  - fmtflags, 1746
  - getloc, 1755
  - good, 1755
  - goodbit, 1781
  - hex, 1781
  - imbue, 1756
  - in, 1781
  - init, 1756
  - internal, 1782
  - iostate, 1747
  - is\_open, 1757
  - isword, 1757
  - left, 1782
  - narrow, 1758
  - oct, 1782
  - open, 1758, 1759
  - openmode, 1747
  - operator bool, 1759
  - operator!, 1759
  - operator<<, 1759–1767
  - out, 1782
  - precision, 1767, 1768
  - put, 1768
  - pword, 1769
  - rdbuf, 1769, 1770
  - rdstate, 1770
  - register\_callback, 1770
  - right, 1783
  - scientific, 1783
  - seekdir, 1747
  - seekp, 1771
  - setf, 1772
  - setstate, 1773
  - showbase, 1783
  - showpoint, 1783
  - showpos, 1784
  - skipws, 1784
  - sync\_with\_stdio, 1773
  - tellp, 1773
  - tie, 1774
  - trunc, 1784
  - unitbuf, 1784
  - unsetf, 1774
  - uppercase, 1785

- widen, 1775
- width, 1775, 1776
- write, 1776
- xalloc, 1777
- std::basic\_ostream< \_CharT, \_Traits >, 1785
  - \_M\_getloc, 1794
  - \_M\_write, 1794
  - \_\_num\_get\_type, 1790
  - ~basic\_ostream, 1794
  - adjustfield, 1821
  - app, 1821
  - ate, 1821
  - bad, 1795
  - badbit, 1821
  - basefield, 1821
  - basic\_ostream, 1794
  - beg, 1822
  - binary, 1822
  - boolalpha, 1822
  - clear, 1795
  - copyfmt, 1795
  - cur, 1822
  - dec, 1823
  - end, 1823
  - eof, 1796
  - eofbit, 1823
  - event, 1793
  - event\_callback, 1791
  - exceptions, 1796
  - fail, 1797
  - failbit, 1823
  - fill, 1797, 1798
  - fixed, 1824
  - flags, 1798, 1799
  - floatfield, 1824
  - flush, 1799
  - fmtflags, 1791
  - getloc, 1799
  - good, 1800
  - goodbit, 1824
  - hex, 1824
  - imbue, 1800
  - in, 1825
  - init, 1801
  - internal, 1825
  - iostate, 1792
  - isword, 1801
  - left, 1825
  - narrow, 1802
  - oct, 1825
  - openmode, 1792
  - operator bool, 1802
  - operator!, 1802
  - operator<<, 1803–1810
  - out, 1826
  - precision, 1810, 1811
  - put, 1811
  - pword, 1812
  - rdbuf, 1812, 1813
  - rdstate, 1813
  - register\_callback, 1814
  - right, 1826
  - scientific, 1826
  - seekdir, 1793
  - seekp, 1814
  - setf, 1815
  - setstate, 1816
  - showbase, 1826
  - showpoint, 1827
  - showpos, 1827
  - skipws, 1827
  - sync\_with\_stdio, 1816
  - tellp, 1817
  - tie, 1817
  - trunc, 1827
  - unitbuf, 1828
  - unsetf, 1818
  - uppercase, 1828
  - widen, 1818
  - width, 1819
  - write, 1819
  - xalloc, 1820
- std::basic\_ostream< \_CharT, \_Traits >::sentry, 3413
  - ~sentry, 3414
  - operator bool, 3415
  - sentry, 3414
- std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1829
  - \_M\_getloc, 1838
  - \_M\_write, 1838
  - \_\_num\_get\_type, 1834
  - ~basic\_ostringstream, 1838
  - adjustfield, 1865
  - app, 1865
  - ate, 1865
  - bad, 1839
  - badbit, 1865
  - basefield, 1866
  - basic\_ostringstream, 1837
  - beg, 1866
  - binary, 1866
  - boolalpha, 1866
  - clear, 1839
  - copyfmt, 1839
  - cur, 1867
  - dec, 1867
  - end, 1867
  - eof, 1840
  - eofbit, 1867



event, 1836  
event\_callback, 1834  
exceptions, 1840  
fail, 1841  
failbit, 1868  
fill, 1841, 1842  
fixed, 1868  
flags, 1842, 1843  
floatfield, 1868  
flush, 1843  
fmtflags, 1834  
getloc, 1843  
good, 1844  
goodbit, 1868  
hex, 1869  
imbue, 1844  
in, 1869  
init, 1845  
internal, 1869  
iostate, 1835  
iword, 1845  
left, 1870  
narrow, 1846  
oct, 1870  
openmode, 1835  
operator bool, 1846  
operator!, 1846  
operator<<, 1847–1854  
out, 1870  
precision, 1854, 1855  
put, 1855  
pword, 1856  
rdbuf, 1856, 1857  
rdstate, 1857  
register\_callback, 1857  
right, 1870  
scientific, 1871  
seekdir, 1836  
seekp, 1858  
setf, 1859  
setstate, 1860  
showbase, 1871  
showpoint, 1871  
showpos, 1871  
skipws, 1872  
str, 1860  
sync\_with\_stdio, 1861  
tellp, 1861  
tie, 1861, 1862  
trunc, 1872  
unitbuf, 1872  
unsetf, 1862  
uppercase, 1872  
widen, 1863  
width, 1863  
write, 1864  
xalloc, 1864  
std::basic\_regex< \_Ch\_type, \_Rx\_traits >, 1873  
    ~basic\_regex, 1879  
    assign, 1879–1882  
    basic\_regex, 1875, 1877, 1878  
    flags, 1883  
    getloc, 1883  
    imbue, 1883  
    mark\_count, 1884  
    operator=, 1884, 1885  
    swap, 1886  
std::basic\_streambuf< \_CharT, \_Traits >, 1887  
    \_M\_buf\_locale, 1908  
    \_M\_in\_beg, 1909  
    \_M\_in\_cur, 1909  
    \_M\_in\_end, 1909  
    \_M\_out\_beg, 1909  
    \_M\_out\_cur, 1909  
    \_M\_out\_end, 1910  
    \_\_streambuf\_type, 1890  
    ~basic\_streambuf, 1892  
    basic\_streambuf, 1892  
    char\_type, 1891  
    eback, 1893  
    egptr, 1893  
    epptr, 1893  
    gbump, 1894  
    getloc, 1894  
    gptr, 1894  
    imbue, 1895  
    in\_avail, 1895  
    int\_type, 1891  
    off\_type, 1891  
    overflow, 1896  
    pbackfail, 1896  
    pbase, 1897  
    pbump, 1897  
    pos\_type, 1891  
    pptr, 1898  
    pubimbue, 1898  
    pubseekoff, 1898  
    pubseekpos, 1899  
    pubsetbuf, 1899  
    pubsync, 1900  
    sbumpc, 1900  
    seekoff, 1900  
    seekpos, 1900  
    setbuf, 1901  
    setg, 1901  
    setp, 1902  
    sgetc, 1902  
    sgetn, 1902

- showmanyc, [1903](#)
- snextc, [1903](#)
- sputbackc, [1904](#)
- sputc, [1904](#)
- sputn, [1905](#)
- sungetc, [1905](#)
- sync, [1906](#)
- traits\_type, [1892](#)
- uflow, [1906](#)
- underflow, [1906](#)
- xsgetn, [1907](#)
- xspn, [1908](#)
- std::basic\_string<\_CharT, \_Traits, \_Alloc >, [1910](#)
  - ~basic\_string, [1919](#)
  - append, [1919–1922](#)
  - assign, [1923–1927](#)
  - at, [1927](#), [1928](#)
  - back, [1928](#), [1929](#)
  - basic\_string, [1915–1919](#)
  - begin, [1929](#)
  - c\_str, [1929](#)
  - capacity, [1929](#)
  - cbegin, [1930](#)
  - cend, [1930](#)
  - clear, [1930](#)
  - compare, [1930–1933](#)
  - copy, [1934](#)
  - crbegin, [1935](#)
  - crend, [1935](#)
  - data, [1935](#)
  - empty, [1935](#)
  - end, [1936](#)
  - erase, [1936](#), [1937](#)
  - find, [1938](#), [1939](#)
  - find\_first\_not\_of, [1940–1942](#)
  - find\_first\_of, [1942–1944](#)
  - find\_last\_not\_of, [1944–1946](#)
  - find\_last\_of, [1946–1948](#)
  - front, [1948](#), [1949](#)
  - get\_allocator, [1949](#)
  - insert, [1949–1954](#)
  - length, [1955](#)
  - max\_size, [1955](#)
  - npos, [1975](#)
  - operator+=, [1955–1957](#)
  - operator=, [1957–1959](#)
  - operator[], [1959](#)
  - pop\_back, [1960](#)
  - push\_back, [1960](#)
  - rbegin, [1961](#)
  - rend, [1961](#)
  - replace, [1961–1969](#)
  - reserve, [1969](#)
  - resize, [1970](#), [1971](#)
  - rfind, [1971–1973](#)
  - shrink\_to\_fit, [1973](#)
  - size, [1973](#)
  - substr, [1974](#)
  - swap, [1974](#)
- std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc >, [2009](#)
  - \_M\_buf\_locale, [2030](#)
  - \_M\_in\_beg, [2030](#)
  - \_M\_in\_cur, [2030](#)
  - \_M\_in\_end, [2030](#)
  - \_M\_mode, [2030](#)
  - \_M\_out\_beg, [2031](#)
  - \_M\_out\_cur, [2031](#)
  - \_M\_out\_end, [2031](#)
- basic\_stringbuf, [2012](#)
- eback, [2013](#)
- egptr, [2013](#)
- eptr, [2013](#)
- gbump, [2014](#)
- getloc, [2014](#)
- gptr, [2014](#)
- imbue, [2015](#)
- in\_avail, [2015](#)
- overflow, [2016](#)
- pbackfail, [2016](#)
- pbase, [2017](#)
- pbump, [2017](#)
- pptr, [2018](#)
- pubimbue, [2018](#)
- pubseekoff, [2018](#)
- pubseekpos, [2019](#)
- pubsetbuf, [2019](#)
- pubsync, [2020](#)
- sbumpc, [2020](#)
- seekoff, [2020](#)
- seekpos, [2020](#)
- setbuf, [2021](#)
- setg, [2021](#)
- setp, [2022](#)
- sgetc, [2022](#)
- sgetn, [2023](#)
- showmanyc, [2023](#)
- snextc, [2024](#)
- sputbackc, [2024](#)
- sputc, [2025](#)
- sputn, [2025](#)
- str, [2026](#)
- sungetc, [2027](#)
- sync, [2027](#)
- uflow, [2027](#)
- underflow, [2028](#)
- xsgetn, [2028](#)
- xspn, [2029](#)
- std::basic\_stringstream<\_CharT, \_Traits, \_Alloc >, [2032](#)

[\\_M\\_gcount](#), [2090](#)  
[\\_M\\_getloc](#), [2043](#)  
[\\_M\\_write](#), [2044](#)  
[\\_\\_num\\_put\\_type](#), [2039](#)  
[~basic\\_stringstream](#), [2043](#)  
[adjustfield](#), [2090](#)  
[app](#), [2090](#)  
[ate](#), [2091](#)  
[bad](#), [2044](#)  
[badbit](#), [2091](#)  
[basefield](#), [2091](#)  
[basic\\_stringstream](#), [2042](#), [2043](#)  
[beg](#), [2091](#)  
[binary](#), [2091](#)  
[boolalpha](#), [2092](#)  
[clear](#), [2045](#)  
[copyfmt](#), [2045](#)  
[cur](#), [2092](#)  
[dec](#), [2092](#)  
[end](#), [2092](#)  
[eof](#), [2046](#)  
[eofbit](#), [2093](#)  
[event](#), [2042](#)  
[event\\_callback](#), [2039](#)  
[exceptions](#), [2046](#)  
[fail](#), [2047](#)  
[failbit](#), [2093](#)  
[fill](#), [2047](#)  
[fixed](#), [2093](#)  
[flags](#), [2048](#)  
[floatfield](#), [2093](#)  
[flush](#), [2049](#)  
[fmtflags](#), [2040](#)  
[gcount](#), [2049](#)  
[get](#), [2049–2052](#)  
[getline](#), [2053](#), [2054](#)  
[getloc](#), [2054](#)  
[good](#), [2054](#)  
[goodbit](#), [2094](#)  
[hex](#), [2094](#)  
[ignore](#), [2055](#), [2056](#)  
[imbue](#), [2056](#)  
[in](#), [2094](#)  
[init](#), [2057](#)  
[internal](#), [2094](#)  
[iostate](#), [2040](#)  
[iword](#), [2057](#)  
[left](#), [2095](#)  
[narrow](#), [2058](#)  
[oct](#), [2095](#)  
[openmode](#), [2041](#)  
[operator bool](#), [2058](#)  
[operator!](#), [2058](#)  
[operator<<](#), [2059–2066](#)  
[operator>>](#), [2066–2074](#)  
[out](#), [2095](#)  
[peek](#), [2074](#)  
[precision](#), [2075](#)  
[put](#), [2076](#)  
[putback](#), [2076](#)  
[pword](#), [2077](#)  
[rdbuf](#), [2077](#), [2078](#)  
[rdstate](#), [2078](#)  
[read](#), [2078](#)  
[readsome](#), [2079](#)  
[register\\_callback](#), [2080](#)  
[right](#), [2095](#)  
[scientific](#), [2096](#)  
[seekdir](#), [2041](#)  
[seekg](#), [2080](#), [2081](#)  
[seekp](#), [2081](#), [2082](#)  
[setf](#), [2082](#), [2083](#)  
[setstate](#), [2083](#)  
[showbase](#), [2096](#)  
[showpoint](#), [2096](#)  
[showpos](#), [2096](#)  
[skipws](#), [2097](#)  
[str](#), [2084](#)  
[sync](#), [2084](#)  
[sync\\_with\\_stdio](#), [2085](#)  
[tellg](#), [2085](#)  
[tellp](#), [2086](#)  
[tie](#), [2086](#)  
[trunc](#), [2097](#)  
[unget](#), [2087](#)  
[unitbuf](#), [2097](#)  
[unsetf](#), [2087](#)  
[uppercase](#), [2097](#)  
[widen](#), [2088](#)  
[width](#), [2088](#)  
[write](#), [2089](#)  
[xalloc](#), [2089](#)  
[std::bernoulli\\_distribution](#), [2098](#)  
[bernoulli\\_distribution](#), [2099](#)  
[max](#), [2100](#)  
[min](#), [2100](#)  
[operator\(\)](#), [2100](#)  
[operator==](#), [2101](#)  
[p](#), [2100](#)  
[param](#), [2101](#)  
[reset](#), [2101](#)  
[result\\_type](#), [2099](#)  
[std::bernoulli\\_distribution::param\\_type](#), [3242](#)  
[std::bidirectional\\_iterator\\_tag](#), [2102](#)  
[std::binary\\_function<\\_Arg1, \\_Arg2, \\_Result >](#), [2119](#)  
[first\\_argument\\_type](#), [2119](#)  
[result\\_type](#), [2119](#)  
[second\\_argument\\_type](#), [2120](#)

std::binary\_negate< \_Predicate >, 2133  
     first\_argument\_type, 2134  
     result\_type, 2134  
     second\_argument\_type, 2134  
 std::binder1st< \_Operation >, 2135  
     argument\_type, 2136  
     result\_type, 2136  
 std::binder2nd< \_Operation >, 2136  
     argument\_type, 2137  
     result\_type, 2137  
 std::binomial\_distribution< \_IntType >, 2138  
     max, 2139  
     min, 2139  
     operator<=, 2142  
     operator>=, 2142  
     operator(), 2140  
     operator==, 2142  
     p, 2140  
     param, 2140, 2141  
     reset, 2141  
     result\_type, 2139  
     t, 2141  
 std::binomial\_distribution< \_IntType >::param\_type, 3237  
 std::bitset< \_Nb >, 2151  
     all, 2157  
     any, 2157  
     bitset, 2155, 2156  
     count, 2157  
     flip, 2157, 2158  
     none, 2158  
     operator!=, 2158  
     operator<=, 2159  
     operator<=, 2159  
     operator>=, 2160  
     operator>=, 2160  
     operator~, 2162  
     operator^=, 2162  
     operator==, 2160  
     operator&=, 2159  
     operator[], 2161  
     operator|=, 2162  
     reset, 2163  
     set, 2163, 2164  
     size, 2164  
     test, 2164  
     to\_string, 2165  
     to\_ulong, 2165  
 std::bitset< \_Nb >::reference, 3340  
 std::cauchy\_distribution< \_RealType >, 2172  
     a, 2173  
     max, 2173  
     min, 2174  
     operator(), 2174  
     operator==, 2175  
     param, 2174  
     reset, 2175  
     result\_type, 2173  
 std::cauchy\_distribution< \_RealType >::param\_type, 3247  
 std::char\_traits< \_\_gnu\_cxx::character< \_Value, \_Int, \_St > >, 2196  
 std::char\_traits< \_CharT >, 2195  
 std::char\_traits< char >, 2197  
 std::char\_traits< wchar\_t >, 2198  
 std::chi\_squared\_distribution< \_RealType >, 2199  
     max, 2201  
     min, 2201  
     n, 2201  
     operator<=, 2203  
     operator>=, 2203  
     operator(), 2201  
     operator==, 2203  
     param, 2202  
     reset, 2202  
     result\_type, 2200  
 std::chi\_squared\_distribution< \_RealType >::param\_type, 3237  
 std::chrono, 824  
 std::chrono::\_V2::steady\_clock, 3546  
 std::chrono::\_V2::system\_clock, 3575  
 std::chrono::duration< \_Rep, \_Period >, 2460  
 std::chrono::duration\_values< \_Rep >, 2461  
 std::chrono::time\_point< \_Clock, \_Dur >, 3622  
 std::chrono::treat\_as\_floating\_point< \_Rep >, 3634  
 std::codecvt< \_InternT, \_ExternT, \_StateT >, 2204  
     do\_out, 2206  
     in, 2206  
     out, 2207  
     unshift, 2208  
 std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2210  
     do\_out, 2211  
     in, 2212  
     out, 2213  
     unshift, 2214  
 std::codecvt< char, char, mbstate\_t >, 2215  
     do\_out, 2216  
     in, 2217  
     out, 2218  
     unshift, 2219  
 std::codecvt< char16\_t, char, mbstate\_t >, 2220  
     do\_out, 2221  
     in, 2222  
     out, 2223  
     unshift, 2224  
 std::codecvt< char32\_t, char, mbstate\_t >, 2225  
     do\_out, 2226  
     in, 2227  
     out, 2228

- unshift, [2229](#)
- std::codecvt< wchar\_t, char, mbstate\_t >, [2230](#)
  - do\_out, [2231](#)
  - in, [2232](#)
  - out, [2233](#)
  - unshift, [2234](#)
- std::codecvt\_base, [2235](#)
- std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, [2236](#)
  - do\_out, [2237](#)
  - in, [2238](#)
  - out, [2239](#)
  - unshift, [2240](#)
- std::collate< \_CharT >, [2241](#)
  - ~collate, [2244](#)
  - char\_type, [2243](#)
  - collate, [2243](#), [2244](#)
  - compare, [2244](#)
  - do\_compare, [2245](#)
  - do\_hash, [2245](#)
  - do\_transform, [2246](#)
  - hash, [2247](#)
  - id, [2248](#)
  - string\_type, [2243](#)
  - transform, [2247](#)
- std::collate\_byname< \_CharT >, [2248](#)
  - char\_type, [2250](#)
  - compare, [2250](#)
  - do\_compare, [2251](#)
  - do\_hash, [2251](#)
  - do\_transform, [2252](#)
  - hash, [2253](#)
  - id, [2254](#)
  - string\_type, [2250](#)
  - transform, [2253](#)
- std::common\_type< \_Tp >, [2254](#)
- std::common\_type< chrono::duration< \_Rep, \_Period >  
>, [2255](#)
- std::common\_type< chrono::duration< \_Rep, \_Period >, chrono::duration< \_Rep, \_Period > >, [2255](#)
- std::common\_type< chrono::duration< \_Rep1, \_Period1 >, chrono::duration< \_Rep2, \_Period2 > >, [2255](#)
- std::common\_type< chrono::time\_point< \_Clock, \_Duration > >, [2256](#)
- std::common\_type< chrono::time\_point< \_Clock, \_Duration >, chrono::time\_point< \_Clock, \_Duration > >, [2256](#)
- std::common\_type< chrono::time\_point< \_Clock, \_Duration1 >, chrono::time\_point< \_Clock, \_Duration2 > >, [2257](#)
- std::complex< \_Tp >, [2257](#)
  - complex, [2259](#)
  - operator+=, [2259](#)
  - operator-=, [2259](#)
  - value\_type, [2258](#)
- std::complex< double >, [2260](#)
- std::complex< float >, [2261](#)
- std::complex< long double >, [2262](#)
- std::condition\_variable, [2265](#)
- std::conditional< \_Cond, \_Iftrue, \_Iffalse >, [2266](#)
- std::const\_mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, [2268](#)
  - first\_argument\_type, [2269](#)
  - result\_type, [2269](#)
  - second\_argument\_type, [2269](#)
- std::const\_mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, [2270](#)
  - first\_argument\_type, [2271](#)
  - result\_type, [2271](#)
  - second\_argument\_type, [2271](#)
- std::const\_mem\_fun\_ref\_t< \_Ret, \_Tp >, [2272](#)
  - argument\_type, [2272](#)
  - result\_type, [2273](#)
- std::const\_mem\_fun\_t< \_Ret, \_Tp >, [2273](#)
  - argument\_type, [2274](#)
  - result\_type, [2274](#)
- std::ctype< \_CharT >, [2301](#)
  - do\_is, [2303](#), [2304](#)
  - do\_narrow, [2304](#), [2305](#)
  - do\_scan\_is, [2306](#)
  - do\_scan\_not, [2306](#)
  - do\_tolower, [2307](#), [2308](#)
  - do\_toupper, [2308](#), [2309](#)
  - do\_widen, [2309](#), [2310](#)
  - id, [2318](#)
  - is, [2311](#)
  - narrow, [2312](#)
  - scan\_is, [2313](#)
  - scan\_not, [2314](#)
  - tolower, [2314](#), [2315](#)
  - toupper, [2315](#), [2316](#)
  - widen, [2316](#), [2318](#)
- std::ctype< char >, [2319](#)
  - ~ctype, [2322](#)
  - char\_type, [2321](#)
  - classic\_table, [2323](#)
  - ctype, [2322](#)
  - do\_narrow, [2323](#)
  - do\_tolower, [2324](#), [2325](#)
  - do\_toupper, [2325](#), [2326](#)
  - do\_widen, [2326](#), [2327](#)
  - id, [2335](#)
  - is, [2327](#), [2328](#)
  - narrow, [2328](#), [2329](#)
  - scan\_is, [2330](#)
  - scan\_not, [2330](#)
  - table, [2331](#)
  - table\_size, [2335](#)
  - tolower, [2331](#), [2332](#)

toupper, 2332, 2333  
 widen, 2334  
 std::ctype< wchar\_t >, 2336  
   ~ctype, 2339  
   char\_type, 2338  
   ctype, 2338  
   do\_is, 2339, 2340  
   do\_narrow, 2340, 2341  
   do\_scan\_is, 2341  
   do\_scan\_not, 2342  
   do\_tolower, 2343  
   do\_toupper, 2344  
   do\_widen, 2346  
   id, 2354  
   is, 2347, 2348  
   narrow, 2348, 2349  
   scan\_is, 2349  
   scan\_not, 2350  
   tolower, 2350, 2351  
   toupper, 2351, 2352  
   widen, 2352, 2353  
 std::ctype\_base, 2354  
 std::ctype\_byname< \_CharT >, 2355  
   do\_is, 2357, 2358  
   do\_narrow, 2358, 2359  
   do\_scan\_is, 2359  
   do\_scan\_not, 2360  
   do\_tolower, 2361  
   do\_toupper, 2362  
   do\_widen, 2363, 2364  
   id, 2372  
   is, 2364, 2365  
   narrow, 2365, 2366  
   scan\_is, 2367  
   scan\_not, 2367  
   tolower, 2368  
   toupper, 2370  
   widen, 2371  
 std::ctype\_byname< char >, 2373  
   char\_type, 2375  
   classic\_table, 2375  
   do\_narrow, 2375, 2376  
   do\_tolower, 2376, 2377  
   do\_toupper, 2377, 2378  
   do\_widen, 2378, 2379  
   id, 2388  
   is, 2380  
   narrow, 2381  
   scan\_is, 2383  
   scan\_not, 2383  
   table, 2384  
   table\_size, 2388  
   tolower, 2384, 2385  
   toupper, 2385, 2386  
   widen, 2387  
 std::decay< \_Tp >, 2390  
 std::decimal, 825  
   decimal32\_to\_long\_long, 834  
 std::decimal::decimal128, 2390  
   decimal128, 2391  
 std::decimal::decimal32, 2392  
   decimal32, 2393  
 std::decimal::decimal64, 2394  
   decimal64, 2395  
 std::default\_delete< \_Tp >, 2396  
   default\_delete, 2396, 2397  
   operator(), 2397  
 std::default\_delete< \_Tp[] >, 2397  
   default\_delete, 2398  
   operator(), 2398  
 std::defer\_lock\_t, 2405  
 std::deque< \_Tp, \_Alloc >, 2405  
   \_M\_fill\_initialize, 2414  
   \_M\_initialize\_map, 2415  
   \_M\_new\_elements\_at\_back, 2415  
   \_M\_new\_elements\_at\_front, 2416  
   \_M\_pop\_back\_aux, 2416  
   \_M\_pop\_front\_aux, 2416  
   \_M\_push\_back\_aux, 2416  
   \_M\_push\_front\_aux, 2416  
   \_M\_range\_check, 2417  
   \_M\_range\_initialize, 2417, 2418  
   \_M\_reallocate\_map, 2418  
   \_M\_reserve\_elements\_at\_back, 2418  
   \_M\_reserve\_elements\_at\_front, 2419  
   \_M\_reserve\_map\_at\_back, 2419  
   \_M\_reserve\_map\_at\_front, 2419  
   ~deque, 2414  
   assign, 2419, 2420  
   at, 2421  
   back, 2422  
   begin, 2422, 2423  
   cbegin, 2423  
   cend, 2423  
   clear, 2423  
   crbegin, 2424  
   crend, 2424  
   deque, 2411–2414  
   emplace, 2424  
   empty, 2425  
   end, 2425  
   erase, 2425, 2426  
   front, 2426  
   get\_allocator, 2427  
   insert, 2427–2429  
   max\_size, 2430  
   operator=, 2430, 2431  
   operator[], 2431, 2432

- pop\_back, [2432](#)
- pop\_front, [2432](#)
- push\_back, [2432](#)
- push\_front, [2433](#)
- rbegin, [2433](#)
- rend, [2434](#)
- resize, [2434](#), [2435](#)
- shrink\_to\_fit, [2435](#)
- size, [2435](#)
- swap, [2435](#)
- std::discard\_block\_engine< \_RandomNumberEngine, \_\_p, \_\_r >, [2444](#)
- base, [2448](#)
- discard, [2448](#)
- discard\_block\_engine, [2446](#), [2447](#)
- max, [2448](#)
- min, [2448](#)
- operator<<, [2450](#)
- operator>>, [2451](#)
- operator(), [2449](#)
- operator==, [2450](#)
- result\_type, [2446](#)
- seed, [2449](#)
- std::discrete\_distribution< \_IntType >, [2451](#)
- max, [2453](#)
- min, [2453](#)
- operator<<, [2455](#)
- operator>>, [2455](#)
- operator(), [2453](#)
- operator==, [2455](#)
- param, [2453](#), [2454](#)
- probabilities, [2454](#)
- reset, [2454](#)
- result\_type, [2453](#)
- std::discrete\_distribution< \_IntType >::param\_type, [3240](#)
- std::divides< \_Tp >, [2456](#)
- first\_argument\_type, [2457](#)
- result\_type, [2457](#)
- second\_argument\_type, [2457](#)
- std::divides< void >, [2458](#)
- std::domain\_error, [2458](#)
- what, [2459](#)
- std::enable\_if< bool, \_Tp >, [2482](#)
- std::enable\_shared\_from\_this< \_Tp >, [2482](#)
- std::equal\_to< \_Tp >, [2515](#)
- first\_argument\_type, [2516](#)
- result\_type, [2516](#)
- second\_argument\_type, [2516](#)
- std::equal\_to< void >, [2517](#)
- std::error\_code, [2518](#)
- std::error\_condition, [2519](#)
- std::exception, [2522](#)
- std::experimental, [834](#)
- std::experimental::filesystem::v1::filesystem\_error, [2537](#)
- what, [2538](#)
- std::experimental::filesystem::v1::path, [3256](#)
- std::experimental::filesystem::v1::path::iterator, [2791](#)
- std::experimental::fundamentals\_v1::any, [1322](#)
- ~any, [1324](#)
- any, [1322](#), [1323](#)
- clear, [1324](#)
- empty, [1324](#)
- operator=, [1324](#), [1325](#)
- swap, [1325](#)
- type, [1325](#)
- std::experimental::fundamentals\_v1::bad\_any\_cast, [1373](#)
- what, [1374](#)
- std::experimental::fundamentals\_v1::bad\_optional\_access, [1377](#)
- what, [1378](#)
- std::experimental::fundamentals\_v1::basic\_string\_view< \_CharT, \_Traits >, [2007](#)
- std::experimental::fundamentals\_v1::in\_place\_t, [2684](#)
- std::experimental::fundamentals\_v1::nullopt\_t, [3110](#)
- std::experimental::fundamentals\_v1::optional< \_Tp >, [3194](#)
- std::experimental::fundamentals\_v2::ostream\_joiner< \_DelimT, \_CharT, \_Traits >, [3200](#)
- std::experimental::fundamentals\_v2::owner\_less< shared\_ptr< \_Tp > >, [3218](#)
- first\_argument\_type, [3219](#)
- result\_type, [3219](#)
- second\_argument\_type, [3219](#)
- std::experimental::fundamentals\_v2::owner\_less< weak\_ptr< \_Tp > >, [3223](#)
- first\_argument\_type, [3224](#)
- result\_type, [3224](#)
- second\_argument\_type, [3224](#)
- std::experimental::fundamentals\_v2::propagate\_const< \_Tp >, [3294](#)
- std::exponential\_distribution< \_RealType >, [2524](#)
- exponential\_distribution, [2525](#), [2526](#)
- lambda, [2526](#)
- max, [2526](#)
- min, [2526](#)
- operator(), [2527](#)
- operator==, [2528](#)
- param, [2527](#)
- reset, [2528](#)
- result\_type, [2525](#)
- std::exponential\_distribution< \_RealType >::param\_type, [3243](#)
- std::extent< typename, \_UInt >, [2528](#)
- std::extreme\_value\_distribution< \_RealType >, [2529](#)
- a, [2531](#)
- b, [2531](#)
- max, [2531](#)
- min, [2531](#)

operator(), 2532  
 operator==, 2533  
 param, 2532  
 reset, 2533  
 result\_type, 2530  
 std::extreme\_value\_distribution<\_RealType>::param\_type, 3239  
 std::fisher\_f\_distribution<\_RealType>, 2539  
   m, 2540  
   max, 2540  
   min, 2541  
   operator<=, 2542  
   operator>=, 2543  
   operator(), 2541  
   operator==, 2542  
   param, 2541  
   reset, 2542  
   result\_type, 2540  
 std::fisher\_f\_distribution<\_RealType>::param\_type, 3241  
 std::forward\_iterator\_tag, 2545  
 std::forward\_list<\_Tp, \_Alloc>, 2550  
   ~forward\_list, 2557  
   assign, 2557, 2558  
   before\_begin, 2559  
   begin, 2559, 2560  
   cbefore\_begin, 2560  
   cbegin, 2560  
   cend, 2560  
   clear, 2561  
   emplace\_after, 2561  
   emplace\_front, 2562  
   empty, 2562  
   end, 2562  
   erase\_after, 2563  
   forward\_list, 2553–2557  
   front, 2564  
   get\_allocator, 2564  
   insert\_after, 2564–2566  
   max\_size, 2567  
   merge, 2567, 2568  
   operator=, 2568, 2569  
   pop\_front, 2569  
   push\_front, 2570  
   remove, 2570  
   remove\_if, 2571  
   resize, 2571, 2572  
   reverse, 2572  
   sort, 2572  
   splice\_after, 2573, 2574  
   swap, 2575  
   unique, 2575  
 std::fpos<\_StateT>, 2576  
   fpos, 2577  
   operator streamoff, 2577  
   operator+, 2577  
   operator+=, 2577  
   operator-, 2578  
   operator=, 2578  
   state, 2578, 2579  
 std::from\_chars\_result, 2581  
 std::front\_insert\_iterator<\_Container>, 2581  
   container\_type, 2582  
   difference\_type, 2582  
   front\_insert\_iterator, 2584  
   iterator\_category, 2583  
   operator\*, 2584  
   operator++, 2584  
   operator=, 2584  
   pointer, 2583  
   reference, 2583  
   value\_type, 2583  
 std::function<\_Res(\_ArgTypes...)>, 2585  
   function, 2587, 2588  
   operator bool, 2589  
   operator(), 2589  
   operator=, 2590–2592  
   swap, 2592  
   target, 2593  
   target\_type, 2593  
 std::future<\_Res>, 2594  
   \_M\_get\_result, 2596  
   \_Ptr, 2596  
   future, 2596  
   get, 2596  
 std::future<\_Res &>, 2597  
   \_M\_get\_result, 2599  
   \_Ptr, 2598  
   future, 2599  
   get, 2599  
 std::future<void>, 2600  
   \_M\_get\_result, 2602  
   \_Ptr, 2601  
   future, 2602  
   get, 2602  
 std::future\_error, 2603  
   what, 2603  
 std::gamma\_distribution<\_RealType>, 2604  
   alpha, 2606  
   beta, 2606  
   gamma\_distribution, 2605  
   max, 2606  
   min, 2606  
   operator<=, 2608  
   operator>=, 2609  
   operator(), 2607  
   operator==, 2608  
   param, 2607



- reset, 2608
- result\_type, 2605
- std::gamma\_distribution< \_RealType >::param\_type, 3246
- std::geometric\_distribution< \_IntType >, 2609
  - max, 2611
  - min, 2611
  - operator(), 2611
  - operator==, 2613
  - p, 2611
  - param, 2612
  - reset, 2612
  - result\_type, 2610
- std::geometric\_distribution< \_IntType >::param\_type, 3246
- std::greater< \_Tp >, 2626
  - first\_argument\_type, 2627
  - result\_type, 2627
  - second\_argument\_type, 2627
- std::greater< void >, 2628
- std::greater\_equal< \_Tp >, 2629
  - first\_argument\_type, 2629
  - result\_type, 2630
  - second\_argument\_type, 2630
- std::greater\_equal< void >, 2630
- std::gslice, 2632
- std::gslice\_array< \_Tp >, 2633
- std::has\_virtual\_destructor< \_Tp >, 2634
- std::hash< \_\_debug::bitset< \_Nb > >, 2635
- std::hash< \_\_debug::vector< bool, \_Alloc > >, 2636
- std::hash< \_\_gnu\_cxx::\_\_u16vstring >, 2637
- std::hash< \_\_gnu\_cxx::\_\_u32vstring >, 2637
- std::hash< \_\_gnu\_cxx::\_\_vstring >, 2638
- std::hash< \_\_gnu\_cxx::\_\_wvstring >, 2638
- std::hash< \_\_gnu\_cxx::throw\_value\_limit >, 2639
  - argument\_type, 2640
  - result\_type, 2640
- std::hash< \_\_gnu\_cxx::throw\_value\_random >, 2641
  - argument\_type, 2641
  - result\_type, 2642
- std::hash< \_\_shared\_ptr< \_Tp, \_Lp > >, 2642
- std::hash< \_Tp >, 2635
- std::hash< \_Tp \* >, 2643
- std::hash< bool >, 2643
- std::hash< char >, 2644
- std::hash< char16\_t >, 2644
- std::hash< char32\_t >, 2645
- std::hash< double >, 2646
- std::hash< error\_code >, 2646
- std::hash< experimental::optional< \_Tp > >, 2647
- std::hash< experimental::shared\_ptr< \_Tp > >, 2647
- std::hash< float >, 2648
- std::hash< int >, 2649
- std::hash< long >, 2649
- std::hash< long double >, 2650
- std::hash< long long >, 2650
- std::hash< shared\_ptr< \_Tp > >, 2651
- std::hash< short >, 2652
- std::hash< signed char >, 2652
- std::hash< string >, 2653
- std::hash< thread::id >, 2653
- std::hash< type\_index >, 2654
- std::hash< u16string >, 2655
- std::hash< u32string >, 2655
- std::hash< unique\_ptr< \_Tp, \_Dp > >, 2656
- std::hash< unsigned char >, 2656
- std::hash< unsigned int >, 2657
- std::hash< unsigned long >, 2658
- std::hash< unsigned long long >, 2658
- std::hash< unsigned short >, 2659
- std::hash< wchar\_t >, 2659
- std::hash< wstring >, 2660
- std::hash<::bitset< \_Nb > >, 2661
- std::hash<::vector< bool, \_Alloc > >, 2661
- std::independent\_bits\_engine< \_RandomNumberEngine, \_\_w, \_UIntType >, 2685
  - base, 2688
  - discard, 2688
  - independent\_bits\_engine, 2686, 2687
  - max, 2688
  - min, 2688
  - operator>>, 2690
  - operator(), 2689
  - operator==, 2690
  - result\_type, 2686
  - seed, 2689
- std::indirect\_array< \_Tp >, 2691
- std::initializer\_list< \_E >, 2693
  - begin, 2693
  - end, 2694
- std::input\_iterator\_tag, 2695
- std::insert\_iterator< \_Container >, 2697
  - container\_type, 2698
  - difference\_type, 2698
  - insert\_iterator, 2699
  - iterator\_category, 2698
  - operator\*, 2700
  - operator++, 2700
  - operator=, 2700
  - pointer, 2699
  - reference, 2699
  - value\_type, 2699
- std::integer\_sequence< \_Tp, \_Idx >, 2701
- std::integral\_constant< \_Tp, \_\_v >, 2702
- std::invalid\_argument, 2703
  - what, 2704
- std::ios\_base, 2704
  - \_M\_getloc, 2710

- ~ios\_base, 2710
- adjustfield, 2717
- app, 2717
- ate, 2717
- badbit, 2717
- basefield, 2717
- beg, 2718
- binary, 2718
- boolalpha, 2718
- cur, 2718
- dec, 2719
- end, 2719
- eofbit, 2719
- event, 2709
- event\_callback, 2707
- failbit, 2719
- fixed, 2720
- flags, 2710
- floatfield, 2720
- fmtflags, 2707
- getloc, 2711
- goodbit, 2720
- hex, 2720
- imbue, 2711
- in, 2721
- internal, 2721
- iostate, 2708
- isword, 2712
- left, 2721
- oct, 2721
- openmode, 2708
- out, 2722
- precision, 2712
- pwd, 2713
- register\_callback, 2713
- right, 2722
- scientific, 2722
- seekdir, 2709
- setf, 2714
- showbase, 2722
- showpoint, 2723
- showpos, 2723
- skipws, 2723
- sync\_with\_stdio, 2715
- trunc, 2723
- unitbuf, 2724
- unsetf, 2715
- uppercase, 2724
- width, 2715, 2716
- xalloc, 2716
- std::ios\_base::failure, 2536
- what, 2537
- std::is\_abstract< \_Tp >, 2724
- std::is\_arithmetic< \_Tp >, 2725
- std::is\_array< typename >, 2726
- std::is\_assignable< \_Tp, \_Up >, 2727
- std::is\_base\_of< \_Base, \_Derived >, 2728
- std::is\_bind\_expression< \_Bind< \_Signature > >, 2730
- std::is\_bind\_expression< \_Bind\_result< \_Result, \_Signature > >, 2731
- std::is\_bind\_expression< \_Tp >, 2729
- std::is\_bind\_expression< const \_Bind< \_Signature > >, 2732
- std::is\_bind\_expression< const \_Bind\_result< \_Result, \_Signature > >, 2733
- std::is\_bind\_expression< const volatile \_Bind< \_Signature > >, 2734
- std::is\_bind\_expression< const volatile \_Bind\_result< \_Result, \_Signature > >, 2735
- std::is\_bind\_expression< volatile \_Bind< \_Signature > >, 2736
- std::is\_bind\_expression< volatile \_Bind\_result< \_Result, \_Signature > >, 2737
- std::is\_class< \_Tp >, 2738
- std::is\_compound< \_Tp >, 2739
- std::is\_const< typename >, 2740
- std::is\_constructible< \_Tp, \_Args >, 2741
- std::is\_convertible< \_From, \_To >, 2741
- std::is\_copy\_assignable< \_Tp >, 2742
- std::is\_copy\_constructible< \_Tp >, 2742
- std::is\_default\_constructible< \_Tp >, 2742
- std::is\_destructible< \_Tp >, 2743
- std::is\_empty< \_Tp >, 2743
- std::is\_enum< \_Tp >, 2744
- std::is\_error\_code\_enum< \_Tp >, 2745
- std::is\_error\_code\_enum< future\_errc >, 2746
- std::is\_error\_condition\_enum< \_Tp >, 2747
- std::is\_final< \_Tp >, 2748
- std::is\_floating\_point< \_Tp >, 2749
- std::is\_function< \_Tp >, 2750
- std::is\_fundamental< \_Tp >, 2751
- std::is\_integral< \_Tp >, 2751
- std::is\_literal\_type< \_Tp >, 2752
- std::is\_lvalue\_reference< typename >, 2753
- std::is\_member\_function\_pointer< \_Tp >, 2754
- std::is\_member\_object\_pointer< \_Tp >, 2755
- std::is\_member\_pointer< \_Tp >, 2755
- std::is\_move\_assignable< \_Tp >, 2756
- std::is\_move\_constructible< \_Tp >, 2757
- std::is\_nothrow\_assignable< \_Tp, \_Up >, 2757
- std::is\_nothrow\_constructible< \_Tp, \_Args >, 2758
- std::is\_nothrow\_copy\_assignable< \_Tp >, 2759
- std::is\_nothrow\_copy\_constructible< \_Tp >, 2759
- std::is\_nothrow\_default\_constructible< \_Tp >, 2759
- std::is\_nothrow\_destructible< \_Tp >, 2760
- std::is\_nothrow\_move\_assignable< \_Tp >, 2761
- std::is\_nothrow\_move\_constructible< \_Tp >, 2761
- std::is\_nothrow\_swappable< \_Tp >, 2761

- std::is\_nothrow\_swappable\_with< \_Tp, \_Up >, 2762
- std::is\_null\_pointer< \_Tp >, 2762
- std::is\_object< \_Tp >, 2763
- std::is\_placeholder< \_Placeholder< \_Num > >, 2765
- std::is\_placeholder< \_Tp >, 2764
- std::is\_pod< \_Tp >, 2766
- std::is\_pointer< \_Tp >, 2767
- std::is\_polymorphic< \_Tp >, 2767
- std::is\_reference< \_Tp >, 2768
- std::is\_rvalue\_reference< typename >, 2769
- std::is\_same< \_Tp, \_Up >, 2770
- std::is\_scalar< \_Tp >, 2771
- std::is\_standard\_layout< \_Tp >, 2771
- std::is\_swappable< \_Tp >, 2772
- std::is\_swappable\_with< \_Tp, \_Up >, 2772
- std::is\_trivial< \_Tp >, 2773
- std::is\_trivially\_assignable< \_Tp, \_Up >, 2774
- std::is\_trivially\_constructible< \_Tp, \_Args >, 2775
- std::is\_trivially\_copy\_assignable< \_Tp >, 2776
- std::is\_trivially\_copy\_constructible< \_Tp >, 2776
- std::is\_trivially\_default\_constructible< \_Tp >, 2777
- std::is\_trivially\_destructible< \_Tp >, 2778
- std::is\_trivially\_move\_assignable< \_Tp >, 2778
- std::is\_trivially\_move\_constructible< \_Tp >, 2778
- std::is\_union< \_Tp >, 2779
- std::is\_void< \_Tp >, 2780
- std::is\_volatile< typename >, 2781
- std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, 2782
  - difference\_type, 2783
  - istream\_iterator, 2784
  - iterator\_category, 2783
  - operator!=, 2785
  - operator==, 2785
  - pointer, 2783
  - reference, 2784
  - value\_type, 2784
- std::istreambuf\_iterator< \_CharT, \_Traits >, 2786
  - char\_type, 2787
  - difference\_type, 2787
  - equal, 2790
  - int\_type, 2787
  - istream\_type, 2788
  - istreambuf\_iterator, 2789, 2790
  - iterator\_category, 2788
  - operator\*, 2790
  - operator++, 2790, 2791
  - pointer, 2788
  - reference, 2788
  - streambuf\_type, 2788
  - traits\_type, 2789
  - value\_type, 2789
- std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, 2793
  - difference\_type, 2794
  - iterator\_category, 2794
  - pointer, 2794
  - reference, 2795
  - value\_type, 2795
- std::iterator\_traits< \_Iterator >, 2795
- std::iterator\_traits< \_Tp \* >, 2796
- std::iterator\_traits< const \_Tp \* >, 2796
- std::length\_error, 2810
  - what, 2811
- std::less< \_Tp >, 2811
  - first\_argument\_type, 2812
  - result\_type, 2812
  - second\_argument\_type, 2812
- std::less< void >, 2813
- std::less\_equal< \_Tp >, 2814
  - first\_argument\_type, 2814
  - result\_type, 2815
  - second\_argument\_type, 2815
- std::less\_equal< void >, 2815
- std::linear\_congruential\_engine< \_UIntType, \_\_a, \_\_c, \_\_m >, 2817
  - discard, 2820
  - increment, 2823
  - linear\_congruential\_engine, 2819
  - max, 2820
  - min, 2820
  - modulus, 2823
  - multiplier, 2824
  - operator<<, 2822
  - operator>>, 2823
  - operator(), 2820
  - operator==, 2822
  - result\_type, 2818
  - seed, 2821
- std::list< \_Tp, \_Alloc >, 2831
  - \_M\_create\_node, 2838
  - ~list, 2838
  - assign, 2838, 2839
  - back, 2840
  - begin, 2840, 2841
  - cbegin, 2841
  - cend, 2841
  - clear, 2841
  - crbegin, 2842
  - crend, 2842
  - emplace, 2842
  - empty, 2843
  - end, 2843
  - erase, 2843, 2844
  - front, 2844, 2845
  - get\_allocator, 2845
  - insert, 2845–2847
  - list, 2835–2837
  - max\_size, 2848

- merge, [2848](#), [2849](#)
- operator=, [2849](#), [2850](#)
- pop\_back, [2851](#)
- pop\_front, [2851](#)
- push\_back, [2851](#)
- push\_front, [2852](#)
- rbegin, [2852](#)
- remove, [2852](#)
- remove\_if, [2853](#)
- rend, [2853](#)
- resize, [2854](#)
- reverse, [2854](#)
- size, [2855](#)
- sort, [2855](#)
- splice, [2855](#)–[2857](#)
- swap, [2858](#)
- unique, [2858](#)
- std::literals::chrono\_literals, [834](#)
  - operator""h, [835](#)
  - operator""min, [835](#), [836](#)
  - operator""ms, [836](#)
  - operator""ns, [836](#)
  - operator""s, [837](#)
  - operator""us, [837](#)
- std::locale, [2861](#)
  - ~locale, [2866](#)
  - all, [2871](#)
  - category, [2863](#)
  - classic, [2867](#)
  - collate, [2871](#)
  - combine, [2867](#)
  - ctype, [2872](#)
  - global, [2867](#)
  - has\_facet, [2870](#)
  - locale, [2863](#)–[2866](#)
  - messages, [2872](#)
  - monetary, [2872](#)
  - name, [2868](#)
  - none, [2872](#)
  - numeric, [2873](#)
  - operator!=, [2868](#)
  - operator(), [2868](#)
  - operator=, [2869](#)
  - operator==, [2869](#)
  - time, [2873](#)
  - use\_facet, [2870](#)
- std::locale::facet, [2534](#)
  - ~facet, [2536](#)
  - facet, [2535](#)
- std::locale::id, [2682](#)
  - has\_facet, [2683](#)
  - id, [2683](#)
  - use\_facet, [2684](#)
- std::lock\_guard< \_Mutex >, [2874](#)
- std::logic\_error, [2874](#)
  - logic\_error, [2875](#)
  - what, [2875](#)
- std::logical\_and< \_Tp >, [2876](#)
  - first\_argument\_type, [2876](#)
  - result\_type, [2877](#)
  - second\_argument\_type, [2877](#)
- std::logical\_and< void >, [2877](#)
- std::logical\_not< \_Tp >, [2878](#)
  - argument\_type, [2879](#)
  - result\_type, [2879](#)
- std::logical\_not< void >, [2879](#)
- std::logical\_or< \_Tp >, [2880](#)
  - first\_argument\_type, [2880](#)
  - result\_type, [2881](#)
  - second\_argument\_type, [2881](#)
- std::logical\_or< void >, [2881](#)
- std::lognormal\_distribution< \_RealType >, [2882](#)
  - m, [2883](#)
  - max, [2883](#)
  - min, [2884](#)
  - operator<=, [2885](#)
  - operator>=, [2886](#)
  - operator(), [2884](#)
  - operator==, [2885](#)
  - param, [2884](#)
  - reset, [2885](#)
  - result\_type, [2883](#)
- std::lognormal\_distribution< \_RealType >::param\_type, [3238](#)
- std::make\_signed< \_Tp >, [2894](#)
- std::make\_unsigned< \_Tp >, [2894](#)
- std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2902](#)
  - ~map, [2909](#)
  - at, [2909](#)
  - begin, [2910](#)
  - cbegin, [2910](#)
  - cend, [2911](#)
  - clear, [2911](#)
  - count, [2911](#), [2912](#)
  - crbegin, [2912](#)
  - crend, [2912](#)
  - emplace, [2913](#)
  - emplace\_hint, [2913](#)
  - empty, [2914](#)
  - end, [2914](#)
  - equal\_range, [2915](#), [2916](#)
  - erase, [2917](#)–[2919](#)
  - find, [2919](#)–[2921](#)
  - get\_allocator, [2921](#)
  - insert, [2921](#)–[2926](#)
  - key\_comp, [2926](#)
  - lower\_bound, [2926](#)–[2928](#)
  - map, [2905](#)–[2908](#)

- max\_size, [2928](#)
- operator=, [2928](#), [2929](#)
- operator[], [2929](#)
- rbegin, [2930](#)
- rend, [2930](#), [2931](#)
- size, [2931](#)
- swap, [2931](#)
- upper\_bound, [2932](#), [2933](#)
- value\_comp, [2934](#)
- std::mask\_array< \_Tp >, [2934](#)
- std::match\_results< \_Bi\_iter, \_Alloc >, [2937](#)
  - ~match\_results, [2942](#)
  - begin, [2942](#)
  - cbegin, [2942](#)
  - cend, [2942](#)
  - empty, [2943](#)
  - end, [2943](#)
  - format, [2943](#), [2944](#)
  - get\_allocator, [2944](#)
  - length, [2945](#)
  - match\_results, [2941](#), [2942](#)
  - max\_size, [2945](#)
  - operator=, [2945](#), [2946](#)
  - operator[], [2946](#)
  - position, [2946](#)
  - prefix, [2947](#)
  - ready, [2947](#)
  - size, [2948](#)
  - str, [2948](#)
  - suffix, [2948](#)
  - swap, [2949](#)
- std::mem\_fun1\_ref\_t< \_Ret, \_Tp, \_Arg >, [2951](#)
  - first\_argument\_type, [2952](#)
  - result\_type, [2952](#)
  - second\_argument\_type, [2952](#)
- std::mem\_fun1\_t< \_Ret, \_Tp, \_Arg >, [2953](#)
  - first\_argument\_type, [2953](#)
  - result\_type, [2954](#)
  - second\_argument\_type, [2954](#)
- std::mem\_fun\_ref\_t< \_Ret, \_Tp >, [2954](#)
  - argument\_type, [2955](#)
  - result\_type, [2955](#)
- std::mem\_fun\_t< \_Ret, \_Tp >, [2956](#)
  - argument\_type, [2956](#)
  - result\_type, [2957](#)
- std::mersenne\_twister\_engine< \_UIntType, \_\_w, \_\_n, \_\_m, \_\_r, \_\_a, \_\_u, \_\_d, \_\_s, \_\_b, \_\_t, \_\_c, \_\_l, \_\_f >, [2957](#)
  - discard, [2960](#)
  - max, [2960](#)
  - mersenne\_twister\_engine, [2960](#)
  - min, [2960](#)
  - operator<<, [2961](#)
  - operator>>, [2962](#)
  - operator==, [2961](#)
  - result\_type, [2959](#)
- std::messages< \_CharT >, [2963](#)
  - ~messages, [2966](#)
  - char\_type, [2965](#)
  - do\_get, [2966](#)
  - id, [2966](#)
  - messages, [2965](#)
  - string\_type, [2965](#)
- std::messages\_base, [2967](#)
- std::messages\_byname< \_CharT >, [2968](#)
  - do\_get, [2969](#)
  - id, [2970](#)
- std::minus< \_Tp >, [2970](#)
  - first\_argument\_type, [2971](#)
  - result\_type, [2971](#)
  - second\_argument\_type, [2971](#)
- std::minus< void >, [2972](#)
- std::modulus< \_Tp >, [2974](#)
  - first\_argument\_type, [2974](#)
  - result\_type, [2975](#)
  - second\_argument\_type, [2975](#)
- std::modulus< void >, [2975](#)
- std::money\_base, [2976](#)
- std::money\_get< \_CharT, \_InIter >, [2977](#)
  - ~money\_get, [2980](#)
  - char\_type, [2979](#)
  - do\_get, [2980](#)
  - get, [2981](#), [2982](#)
  - id, [2983](#)
  - iter\_type, [2979](#)
  - money\_get, [2979](#)
  - string\_type, [2979](#)
- std::money\_put< \_CharT, \_OutIter >, [2983](#)
  - ~money\_put, [2986](#)
  - char\_type, [2985](#)
  - do\_put, [2986](#), [2987](#)
  - id, [2989](#)
  - iter\_type, [2985](#)
  - money\_put, [2985](#)
  - put, [2987](#), [2988](#)
  - string\_type, [2985](#)
- std::moneypunct< \_CharT, \_Intl >, [2990](#)
  - ~moneypunct, [2993](#)
  - char\_type, [2992](#)
  - curr\_symbol, [2994](#)
  - decimal\_point, [2994](#)
  - do\_curr\_symbol, [2994](#)
  - do\_decimal\_point, [2995](#)
  - do\_frac\_digits, [2995](#)
  - do\_grouping, [2996](#)
  - do\_neg\_format, [2996](#)
  - do\_negative\_sign, [2997](#)
  - do\_pos\_format, [2997](#)

- do\_positive\_sign, 2998
- do\_thousands\_sep, 2998
- frac\_digits, 2999
- grouping, 2999
- id, 3002
- intl, 3003
- moneypunct, 2992, 2993
- neg\_format, 3000
- negative\_sign, 3000
- pos\_format, 3001
- positive\_sign, 3001
- string\_type, 2992
- thousands\_sep, 3002
- std::moneypunct\_byname< \_CharT, \_Intl >, 3003
  - curr\_symbol, 3005
  - decimal\_point, 3006
  - do\_curr\_symbol, 3006
  - do\_decimal\_point, 3006
  - do\_frac\_digits, 3007
  - do\_grouping, 3007
  - do\_neg\_format, 3008
  - do\_negative\_sign, 3008
  - do\_pos\_format, 3009
  - do\_positive\_sign, 3009
  - do\_thousands\_sep, 3010
  - frac\_digits, 3010
  - grouping, 3011
  - id, 3014
  - neg\_format, 3011
  - negative\_sign, 3012
  - pos\_format, 3012
  - positive\_sign, 3013
  - thousands\_sep, 3013
- std::move\_iterator< \_Iterator >, 3014
- std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3021
  - ~multimap, 3028
  - begin, 3029
  - cbegin, 3029
  - cend, 3029
  - clear, 3030
  - count, 3030
  - crbegin, 3031
  - crend, 3031
  - emplace, 3031
  - emplace\_hint, 3032
  - empty, 3032
  - end, 3033
  - equal\_range, 3033–3035
  - erase, 3035–3037
  - find, 3038, 3039
  - get\_allocator, 3040
  - insert, 3040–3044
  - key\_comp, 3044
  - lower\_bound, 3045, 3046
  - max\_size, 3047
  - multimap, 3025–3028
  - operator=, 3047
  - rbegin, 3048
  - rend, 3048
  - size, 3049
  - swap, 3049
  - upper\_bound, 3049–3051
  - value\_comp, 3051
- std::multiplies< \_Tp >, 3052
  - first\_argument\_type, 3052
  - result\_type, 3053
  - second\_argument\_type, 3053
- std::multiplies< void >, 3053
- std::multiset< \_Key, \_Compare, \_Alloc >, 3054
  - ~multiset, 3061
  - begin, 3061
  - cbegin, 3061
  - cend, 3061
  - clear, 3062
  - count, 3062
  - crbegin, 3063
  - crend, 3063
  - emplace, 3063
  - emplace\_hint, 3064
  - empty, 3064
  - end, 3065
  - equal\_range, 3065–3067
  - erase, 3067, 3068
  - find, 3069, 3070
  - get\_allocator, 3071
  - insert, 3071–3073
  - key\_comp, 3073
  - lower\_bound, 3073–3075
  - max\_size, 3075
  - multiset, 3057–3060
  - operator=, 3075, 3076
  - rbegin, 3076
  - rend, 3077
  - size, 3077
  - swap, 3077
  - upper\_bound, 3078, 3079
  - value\_comp, 3079
- std::mutex, 3090
- std::negate< \_Tp >, 3091
  - argument\_type, 3091
  - result\_type, 3091
- std::negate< void >, 3092
- std::negative\_binomial\_distribution< \_IntType >, 3092
  - k, 3094
  - max, 3094
  - min, 3094
  - operator<<, 3096
  - operator>>, 3096

- operator(), 3094
- operator==, 3096
- p, 3095
- param, 3095
- reset, 3096
- result\_type, 3094
- std::negative\_binomial\_distribution< \_IntType >::param\_type, 3245
- std::nested\_exception, 3097
- std::normal\_distribution< \_RealType >, 3101
  - max, 3103
  - mean, 3103
  - min, 3103
  - normal\_distribution, 3102
  - operator<=, 3105
  - operator>=, 3106
  - operator(), 3103, 3104
  - operator==, 3106
  - param, 3104
  - reset, 3105
  - result\_type, 3102
  - stddev, 3105
- std::normal\_distribution< \_RealType >::param\_type, 3244
- std::not\_equal\_to< \_Tp >, 3107
  - first\_argument\_type, 3107
  - result\_type, 3108
  - second\_argument\_type, 3108
- std::not\_equal\_to< void >, 3108
- std::num\_get< \_CharT, \_InIter >, 3111
  - ~num\_get, 3114
  - char\_type, 3113
  - do\_get, 3114–3121
  - get, 3122, 3124–3131
  - id, 3132
  - iter\_type, 3113
  - num\_get, 3114
- std::num\_put< \_CharT, \_OutIter >, 3133
  - ~num\_put, 3135
  - char\_type, 3135
  - do\_put, 3136–3140
  - id, 3149
  - iter\_type, 3135
  - num\_put, 3135
  - put, 3141–3148
- std::numeric\_limits< \_Tp >, 3149
  - denorm\_min, 3151
  - digits, 3153
  - digits10, 3153
  - epsilon, 3151
  - has\_denorm, 3153
  - has\_denorm\_loss, 3153
  - has\_infinity, 3153
  - has\_quiet\_NaN, 3154
  - has\_signaling\_NaN, 3154
  - infinity, 3151
  - is\_bounded, 3154
  - is\_exact, 3154
  - is\_iec559, 3154
  - is\_integer, 3155
  - is\_modulo, 3155
  - is\_signed, 3155
  - is\_specialized, 3155
  - lowest, 3151
  - max, 3151
  - max\_digits10, 3155
  - max\_exponent, 3156
  - max\_exponent10, 3156
  - min, 3152
  - min\_exponent, 3156
  - min\_exponent10, 3156
  - quiet\_NaN, 3152
  - radix, 3156
  - round\_error, 3152
  - round\_style, 3157
  - signaling\_NaN, 3152
  - tinyness\_before, 3157
  - traps, 3157
- std::numeric\_limits< bool >, 3158
- std::numeric\_limits< char >, 3159
- std::numeric\_limits< char16\_t >, 3160
- std::numeric\_limits< char32\_t >, 3161
- std::numeric\_limits< double >, 3162
- std::numeric\_limits< float >, 3163
- std::numeric\_limits< int >, 3164
- std::numeric\_limits< long >, 3165
- std::numeric\_limits< long double >, 3166
- std::numeric\_limits< long long >, 3167
- std::numeric\_limits< short >, 3168
- std::numeric\_limits< signed char >, 3169
- std::numeric\_limits< unsigned char >, 3170
- std::numeric\_limits< unsigned int >, 3171
- std::numeric\_limits< unsigned long >, 3172
- std::numeric\_limits< unsigned long long >, 3173
- std::numeric\_limits< unsigned short >, 3174
- std::numeric\_limits< wchar\_t >, 3175
- std::numprint< \_CharT >, 3176
  - ~numprint, 3179
  - char\_type, 3177
  - decimal\_point, 3179
  - do\_decimal\_point, 3180
  - do\_falsename, 3180
  - do\_grouping, 3180
  - do\_thousands\_sep, 3181
  - do\_truename, 3181
  - falsename, 3182
  - grouping, 3182
  - id, 3184



- numpunct, 3178, 3179
- string\_type, 3178
- thousands\_sep, 3183
- truename, 3183
- std::numpunct\_byname< \_CharT >, 3184
  - decimal\_point, 3186
  - do\_decimal\_point, 3186
  - do\_falsename, 3186
  - do\_grouping, 3187
  - do\_thousands\_sep, 3187
  - do\_truename, 3188
  - falsename, 3188
  - grouping, 3188
  - id, 3190
  - thousands\_sep, 3189
  - truename, 3189
- std::once\_flag, 3193
  - call\_once, 3194
  - once\_flag, 3193
  - operator=, 3194
- std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, 3196
  - char\_type, 3197
  - difference\_type, 3198
  - iterator\_category, 3198
  - operator=, 3200
  - ostream\_iterator, 3199, 3200
  - ostream\_type, 3198
  - pointer, 3198
  - reference, 3198
  - traits\_type, 3199
  - value\_type, 3199
- std::ostreambuf\_iterator< \_CharT, \_Traits >, 3201
  - char\_type, 3202
  - difference\_type, 3203
  - failed, 3205
  - iterator\_category, 3203
  - operator\*, 3205
  - operator++, 3205, 3206
  - operator=, 3206
  - ostream\_type, 3203
  - ostreambuf\_iterator, 3204, 3205
  - pointer, 3203
  - reference, 3203
  - streambuf\_type, 3204
  - traits\_type, 3204
  - value\_type, 3204
- std::out\_of\_range, 3207
  - what, 3207
- std::output\_iterator\_tag, 3208
- std::overflow\_error, 3217
  - what, 3217
- std::owner\_less< \_Tp >, 3218
- std::owner\_less< shared\_ptr< \_Tp > >, 3219
  - first\_argument\_type, 3220
  - result\_type, 3220
  - second\_argument\_type, 3220
- std::owner\_less< void >, 3221
  - first\_argument\_type, 3221
  - result\_type, 3222
  - second\_argument\_type, 3222
- std::owner\_less< weak\_ptr< \_Tp > >, 3222
  - first\_argument\_type, 3223
  - result\_type, 3223
  - second\_argument\_type, 3223
- std::packaged\_task< \_Res(\_ArgTypes...)>, 3225
- std::pair< \_T1, \_T2 >, 3226
  - first, 3230
  - first\_type, 3228
  - pair, 3228, 3229
  - second, 3230
  - second\_type, 3228
  - swap, 3230
- std::piecewise\_constant\_distribution< \_RealType >, 3258
  - densities, 3260
  - intervals, 3260
  - max, 3260
  - min, 3260
  - operator<=, 3262
  - operator>=, 3262
  - operator(), 3260
  - operator==, 3262
  - param, 3261
  - reset, 3261
  - result\_type, 3259
- std::piecewise\_constant\_distribution< \_RealType >::param\_type, 3249
- std::piecewise\_construct\_t, 3263
- std::piecewise\_linear\_distribution< \_RealType >, 3263
  - densities, 3265
  - intervals, 3265
  - max, 3265
  - min, 3266
  - operator<=, 3267
  - operator>=, 3268
  - operator(), 3266
  - operator==, 3267
  - param, 3266
  - reset, 3267
  - result\_type, 3265
- std::piecewise\_linear\_distribution< \_RealType >::param\_type, 3241
- std::placeholders, 838
- std::plus< \_Tp >, 3269
  - first\_argument\_type, 3269
  - result\_type, 3270
  - second\_argument\_type, 3270
- std::pointer\_to\_binary\_function< \_Arg1, \_Arg2, \_Result >, 3271



first\_argument\_type, [3272](#)  
result\_type, [3272](#)  
second\_argument\_type, [3272](#)  
std::pointer\_to\_unary\_function< \_Arg, \_Result >, [3273](#)  
argument\_type, [3273](#)  
result\_type, [3274](#)  
std::pointer\_traits< \_Ptr >, [3274](#)  
difference\_type, [3275](#)  
element\_type, [3275](#)  
pointer, [3275](#)  
rebind, [3275](#)  
std::pointer\_traits< \_Tp \* >, [3276](#)  
difference\_type, [3276](#)  
element\_type, [3276](#)  
pointer, [3276](#)  
pointer\_to, [3277](#)  
std::poisson\_distribution< \_IntType >, [3277](#)  
max, [3279](#)  
mean, [3279](#)  
min, [3279](#)  
operator<=, [3281](#)  
operator>=, [3282](#)  
operator(), [3279](#), [3280](#)  
operator==, [3281](#)  
param, [3280](#)  
reset, [3281](#)  
result\_type, [3279](#)  
std::poisson\_distribution< \_IntType >::param\_type, [3236](#)  
std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3284](#)  
empty, [3286](#)  
pop, [3287](#)  
priority\_queue, [3285](#), [3286](#)  
push, [3287](#)  
size, [3287](#)  
top, [3288](#)  
std::promise< \_Res >, [3292](#)  
std::promise< \_Res & >, [3293](#)  
std::promise< void >, [3293](#)  
std::queue< \_Tp, \_Sequence >, [3296](#)  
back, [3298](#)  
c, [3301](#)  
empty, [3299](#)  
front, [3299](#)  
pop, [3299](#)  
push, [3300](#)  
queue, [3298](#)  
size, [3301](#)  
std::random\_access\_iterator\_tag, [3303](#)  
std::random\_device, [3304](#)  
result\_type, [3305](#)  
std::range\_error, [3306](#)  
what, [3306](#)  
std::rank< typename >, [3314](#)  
std::ratio< \_Num, \_Den >, [3315](#)  
std::ratio\_equal< \_R1, \_R2 >, [3316](#)  
std::ratio\_greater< \_R1, \_R2 >, [3317](#)  
std::ratio\_greater\_equal< \_R1, \_R2 >, [3318](#)  
std::ratio\_less< \_R1, \_R2 >, [3319](#)  
std::ratio\_less\_equal< \_R1, \_R2 >, [3319](#)  
std::ratio\_not\_equal< \_R1, \_R2 >, [3320](#)  
std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3321](#)  
difference\_type, [3322](#)  
iterator\_category, [3322](#)  
pointer, [3323](#)  
reference, [3323](#)  
value\_type, [3323](#)  
std::recursive\_mutex, [3339](#)  
std::recursive\_timed\_mutex, [3339](#)  
std::reference\_wrapper< \_Tp >, [3341](#)  
cref, [3342](#)  
ref, [3343](#)  
std::regex\_constants, [838](#)  
\_\_match\_flag, [840](#)  
\_\_polynomial, [848](#)  
\_\_syntax\_option, [840](#)  
awk, [848](#)  
basic, [848](#)  
collate, [849](#)  
ECMAScript, [849](#)  
egrep, [849](#)  
error\_backref, [842](#)  
error\_badbrace, [842](#)  
error\_badrepeat, [842](#)  
error\_brace, [842](#)  
error\_brack, [842](#)  
error\_collate, [842](#)  
error\_complexity, [842](#)  
error\_ctype, [843](#)  
error\_escape, [843](#)  
error\_paren, [843](#)  
error\_range, [843](#)  
error\_space, [843](#)  
error\_stack, [843](#)  
error\_type, [841](#)  
extended, [849](#)  
format\_default, [850](#)  
format\_first\_only, [850](#)  
format\_no\_copy, [850](#)  
format\_sed, [851](#)  
grep, [851](#)  
icase, [851](#)  
match\_any, [851](#)  
match\_continuous, [851](#)  
match\_default, [852](#)  
match\_flag\_type, [841](#)  
match\_not\_bol, [852](#)  
match\_not\_bow, [852](#)  
match\_not\_eol, [852](#)

- match\_not\_eow, [852](#)
- match\_not\_null, [853](#)
- match\_prev\_avail, [853](#)
- nosubs, [853](#)
- operator~, [847](#), [848](#)
- operator^, [845](#)
- operator^=, [845](#), [846](#)
- operator&, [843](#), [844](#)
- operator&=, [844](#)
- operator|, [846](#)
- operator|=, [847](#)
- optimize, [853](#)
- syntax\_option\_type, [841](#)
- std::regex\_error, [3344](#)
  - code, [3345](#)
  - regex\_error, [3344](#)
  - what, [3345](#)
- std::regex\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3345](#)
  - operator!=, [3347](#)
  - operator\*, [3347](#)
  - operator++, [3348](#)
  - operator->, [3348](#)
  - operator=, [3348](#)
  - operator==, [3349](#)
  - regex\_iterator, [3346](#), [3347](#)
- std::regex\_token\_iterator< \_Bi\_iter, \_Ch\_type, \_Rx\_traits >, [3349](#)
  - operator!=, [3353](#)
  - operator\*, [3353](#)
  - operator++, [3353](#)
  - operator->, [3354](#)
  - operator=, [3354](#)
  - operator==, [3354](#)
  - regex\_token\_iterator, [3350–3352](#)
- std::regex\_traits< \_Ch\_type >, [3355](#)
  - getloc, [3356](#)
  - imbue, [3356](#)
  - isctype, [3357](#)
  - length, [3357](#)
  - lookup\_classname, [3358](#)
  - lookup\_collatename, [3359](#)
  - regex\_traits, [3356](#)
  - transform, [3359](#)
  - transform\_primary, [3360](#)
  - translate, [3360](#)
  - translate\_nocase, [3362](#)
  - value, [3362](#)
- std::rel\_ops, [854](#)
  - operator!=, [854](#)
  - operator<=, [854](#)
  - operator>, [855](#)
  - operator>=, [855](#)
- std::remove\_all\_extents< \_Tp >, [3363](#)
- std::remove\_const< \_Tp >, [3363](#)
- std::remove\_cv< \_Tp >, [3364](#)
- std::remove\_extent< \_Tp >, [3364](#)
- std::remove\_pointer< \_Tp >, [3365](#)
- std::remove\_reference< \_Tp >, [3365](#)
- std::remove\_volatile< \_Tp >, [3366](#)
- std::result\_of< \_Signature >, [3369](#)
- std::reverse\_iterator< \_Iterator >, [3369](#)
  - base, [3372](#)
  - iterator\_category, [3371](#)
  - operator\*, [3372](#)
  - operator+, [3373](#)
  - operator++, [3373](#)
  - operator+=, [3374](#)
  - operator-, [3374](#)
  - operator->, [3375](#)
  - operator--, [3374](#), [3375](#)
  - operator=, [3375](#)
  - operator[], [3376](#)
  - reverse\_iterator, [3371](#), [3372](#)
  - value\_type, [3371](#)
- std::runtime\_error, [3381](#)
  - runtime\_error, [3382](#)
  - what, [3382](#)
- std::scoped\_allocator\_adaptor< \_OuterAlloc, \_InnerAllocs >, [3405](#)
- std::seed\_seq, [3407](#)
  - result\_type, [3407](#)
  - seed\_seq, [3408](#)
- std::set< \_Key, \_Compare, \_Alloc >, [3416](#)
  - ~set, [3427](#)
  - allocator\_type, [3419](#)
  - begin, [3427](#)
  - cbegin, [3427](#)
  - cend, [3427](#)
  - clear, [3428](#)
  - const\_iterator, [3419](#)
  - const\_pointer, [3420](#)
  - const\_reference, [3420](#)
  - const\_reverse\_iterator, [3420](#)
  - count, [3428](#)
  - crbegin, [3429](#)
  - crend, [3429](#)
  - difference\_type, [3420](#)
  - emplace, [3429](#)
  - emplace\_hint, [3430](#)
  - empty, [3430](#)
  - end, [3431](#)
  - equal\_range, [3431–3433](#)
  - erase, [3433](#), [3434](#)
  - find, [3435](#), [3436](#)
  - get\_allocator, [3437](#)
  - insert, [3437–3439](#)
  - iterator, [3421](#)

- key\_comp, 3439
- key\_compare, 3421
- key\_type, 3421
- lower\_bound, 3439–3441
- max\_size, 3441
- operator=, 3441, 3442
- pointer, 3421
- rbegin, 3442
- reference, 3422
- rend, 3443
- reverse\_iterator, 3422
- set, 3423–3426
- size, 3443
- size\_type, 3422
- swap, 3443
- upper\_bound, 3444, 3445
- value\_comp, 3445
- value\_compare, 3422
- value\_type, 3423
- std::shared\_future< \_Res >, 3452
  - \_M\_get\_result, 3454
  - \_Ptr, 3453
  - get, 3454
  - shared\_future, 3454
- std::shared\_future< \_Res & >, 3455
  - \_M\_get\_result, 3458
  - \_Ptr, 3457
  - get, 3458
  - shared\_future, 3457
- std::shared\_future< void >, 3458
  - \_M\_get\_result, 3461
  - \_Ptr, 3460
  - shared\_future, 3460
- std::shared\_lock< \_Mutex >, 3461
- std::shared\_ptr< \_Tp >, 3462
  - element\_type, 3466
  - get, 3473
  - operator bool, 3473
  - owner\_before, 3473, 3474
  - shared\_ptr, 3466–3473
  - swap, 3474
  - unique, 3474
  - use\_count, 3474
- std::shared\_timed\_mutex, 3475
- std::shuffle\_order\_engine< \_RandomNumberEngine, \_\_k >, 3476
  - base, 3479
  - discard, 3479
  - max, 3479
  - min, 3479
  - operator<<, 3481
  - operator>>, 3482
  - operator(), 3480
  - operator==, 3481
  - result\_type, 3477
  - seed, 3480
  - shuffle\_order\_engine, 3477, 3478
- std::slice, 3482
- std::slice\_array< \_Tp >, 3483
- std::stack< \_Tp, \_Sequence >, 3493
  - empty, 3495
  - pop, 3495
  - push, 3495
  - size, 3495
  - stack, 3494
  - top, 3496
- std::student\_t\_distribution< \_RealType >, 3551
  - max, 3553
  - min, 3553
  - n, 3553
  - operator<<, 3555
  - operator>>, 3555
  - operator(), 3553
  - operator==, 3555
  - param, 3554
  - reset, 3554
  - result\_type, 3553
- std::student\_t\_distribution< \_RealType >::param\_type, 3239
- std::sub\_match< \_Biter >, 3556
  - compare, 3560, 3561
  - first, 3566
  - first\_type, 3560
  - length, 3562
  - make\_pair, 3563
  - operator string\_type, 3562
  - operator!=, 3564
  - operator<, 3564
  - operator<=, 3564
  - operator>, 3565
  - operator>=, 3565
  - operator==, 3564
  - second, 3566
  - second\_type, 3560
  - str, 3562
  - swap, 3563, 3565
- std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, 3566
  - discard, 3568
  - max, 3569
  - min, 3569
  - operator<<, 3570
  - operator>>, 3571
  - operator(), 3569
  - operator==, 3571
  - result\_type, 3567
  - seed, 3569, 3570
  - subtract\_with\_carry\_engine, 3568

std::system\_error, 3576  
     what, 3577  
 std::this\_thread, 856  
     get\_id, 856  
     sleep\_for, 857  
     sleep\_until, 857  
     yield, 857  
 std::thread, 3583  
     native\_handle, 3584  
 std::thread::id, 2682  
 std::time\_base, 3593  
 std::time\_get<\_CharT, \_InIter>, 3594  
     ~time\_get, 3597  
     char\_type, 3596  
     date\_order, 3597  
     do\_date\_order, 3597  
     do\_get, 3598  
     do\_get\_date, 3599  
     do\_get\_monthname, 3599  
     do\_get\_time, 3600  
     do\_get\_weekday, 3601  
     do\_get\_year, 3602  
     get, 3602, 3603  
     get\_date, 3604  
     get\_monthname, 3605  
     get\_time, 3606  
     get\_weekday, 3606  
     get\_year, 3607  
     id, 3608  
     iter\_type, 3596  
     time\_get, 3596  
 std::time\_get\_byname<\_CharT, \_InIter>, 3609  
     date\_order, 3611  
     do\_date\_order, 3611  
     do\_get, 3611  
     do\_get\_date, 3612  
     do\_get\_monthname, 3613  
     do\_get\_time, 3614  
     do\_get\_weekday, 3615  
     do\_get\_year, 3615  
     get, 3616, 3617  
     get\_date, 3618  
     get\_monthname, 3618  
     get\_time, 3619  
     get\_weekday, 3620  
     get\_year, 3621  
     id, 3621  
 std::time\_put<\_CharT, \_OutIter>, 3623  
     ~time\_put, 3625  
     char\_type, 3624  
     do\_put, 3626  
     id, 3628  
     iter\_type, 3625  
     put, 3626, 3627  
     time\_put, 3625  
 std::time\_put\_byname<\_CharT, \_OutIter>, 3629  
     do\_put, 3630  
     id, 3632  
     put, 3631  
 std::timed\_mutex, 3633  
 std::to\_chars\_result, 3633  
 std::tr1, 857  
 std::tr1::\_\_detail, 860  
 std::tr2, 860  
 std::tr2::\_\_detail, 862  
 std::tr2::\_\_dynamic\_bitset\_base<\_WordT, \_Alloc>, 917  
     \_M\_w, 919  
 std::tr2::\_\_reflection\_typelist<\_Elements>, 964  
 std::tr2::\_\_reflection\_typelist<\_First, \_Rest...>, 964  
 std::tr2::\_\_reflection\_typelist<>, 964  
 std::tr2::bases<\_Tp>, 1383  
 std::tr2::bool\_set, 2167  
     bool\_set, 2168  
     equals, 2169  
     is\_emptyset, 2169  
     is\_indeterminate, 2169  
     is\_singleton, 2169  
     operator bool, 2169  
 std::tr2::direct\_bases<\_Tp>, 2441  
 std::tr2::dynamic\_bitset<\_WordT, \_Alloc>, 2462  
     all, 2469  
     any, 2469  
     append, 2469, 2470  
     clear, 2470  
     count, 2470  
     dynamic\_bitset, 2466–2468  
     empty, 2470  
     find\_first, 2471  
     find\_next, 2471  
     flip, 2472  
     get\_allocator, 2472  
     max\_size, 2472  
     none, 2473  
     num\_blocks, 2473  
     operator<<, 2474  
     operator<=, 2474  
     operator>>, 2475  
     operator>=, 2475  
     operator~, 2477  
     operator^=, 2477  
     operator-=, 2474  
     operator=, 2475  
     operator&=, 2473, 2474  
     operator[], 2476  
     operator|=, 2477  
     push\_back, 2478  
     reset, 2478  
     resize, 2479

- set, [2479](#)
- size, [2480](#)
- swap, [2480](#)
- test, [2480](#)
- to\_string, [2481](#)
- to\_ullong, [2481](#)
- to\_ulong, [2481](#)
- std::tr2::dynamic\_bitset< \_WordT, \_Alloc >::reference, [3341](#)
- std::try\_to\_lock\_t, [3671](#)
- std::tuple< \_Elements >, [3672](#)
- std::tuple< \_T1, \_T2 >, [3674](#)
- std::tuple\_element< 0, std::pair< \_Tp1, \_Tp2 > >, [3676](#)
- std::tuple\_element< 0, tuple< \_Head, \_Tail... > >, [3677](#)
- std::tuple\_element< 1, std::pair< \_Tp1, \_Tp2 > >, [3677](#)
- std::tuple\_element< \_\_i, tuple< \_Head, \_Tail... > >, [3678](#)
- std::tuple\_element< \_\_i, tuple< > >, [3678](#)
- std::tuple\_element< \_Int, ::array< \_Tp, \_Nm > >, [3679](#)
- std::tuple\_element< \_Int, \_Tp >, [3676](#)
- std::tuple\_element< \_Int, std::\_\_debug::array< \_Tp, \_Nm > >, [3679](#)
- std::tuple\_size< \_Tp >, [3679](#)
- std::tuple\_size< std::\_\_debug::array< \_Tp, \_Nm > >, [3680](#)
- std::tuple\_size< std::pair< \_Tp1, \_Tp2 > >, [3681](#)
- std::tuple\_size< tuple< \_Elements... > >, [3682](#)
- std::tuple\_size< ::array< \_Tp, \_Nm > >, [3683](#)
- std::type\_index, [3685](#)
- std::type\_info, [3685](#)
  - ~type\_info, [3686](#)
  - name, [3686](#)
- std::unary\_function< \_Arg, \_Result >, [3689](#)
  - argument\_type, [3690](#)
  - result\_type, [3690](#)
- std::unary\_negate< \_Predicate >, [3691](#)
  - argument\_type, [3691](#)
  - result\_type, [3692](#)
- std::underflow\_error, [3694](#)
  - what, [3694](#)
- std::underlying\_type< \_Tp >, [3695](#)
- std::uniform\_int\_distribution< \_IntType >, [3695](#)
  - max, [3697](#)
  - min, [3697](#)
  - operator(), [3697](#)
  - operator==, [3699](#)
  - param, [3698](#)
  - reset, [3698](#)
  - result\_type, [3696](#)
  - uniform\_int\_distribution, [3697](#)
- std::uniform\_int\_distribution< \_IntType >::param\_type, [3248](#)
- std::uniform\_real\_distribution< \_RealType >, [3699](#)
  - max, [3701](#)
  - min, [3701](#)
- operator(), [3701](#)
- operator==, [3702](#)
- param, [3701](#), [3702](#)
- reset, [3702](#)
- result\_type, [3700](#)
- uniform\_real\_distribution, [3700](#)
- std::uniform\_real\_distribution< \_RealType >::param\_type, [3248](#)
- std::unique\_lock< \_Mutex >, [3703](#)
  - swap, [3704](#)
- std::unique\_ptr< \_Tp, \_Dp >, [3704](#)
  - ~unique\_ptr, [3709](#)
  - get, [3709](#)
  - get\_deleter, [3709](#), [3710](#)
  - operator bool, [3710](#)
  - operator\*, [3710](#)
  - operator->, [3710](#)
  - operator=, [3711](#)
  - release, [3711](#)
  - reset, [3712](#)
  - swap, [3712](#)
  - unique\_ptr, [3707](#), [3708](#)
- std::unique\_ptr< \_Tp[], \_Dp >, [3713](#)
  - ~unique\_ptr, [3716](#)
  - get, [3716](#)
  - get\_deleter, [3716](#), [3717](#)
  - operator bool, [3717](#)
  - operator=, [3717](#), [3718](#)
  - operator[], [3718](#)
  - release, [3718](#)
  - reset, [3718](#)
  - swap, [3719](#)
  - unique\_ptr, [3714](#)–[3716](#)
- std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3719](#)
  - allocator\_type, [3723](#)
  - at, [3730](#)
  - begin, [3731](#), [3732](#)
  - bucket\_count, [3732](#)
  - cbegin, [3732](#)
  - cend, [3733](#)
  - clear, [3734](#)
  - const\_iterator, [3723](#)
  - const\_local\_iterator, [3723](#)
  - const\_pointer, [3724](#)
  - const\_reference, [3724](#)
  - count, [3734](#)
  - difference\_type, [3724](#)
  - emplace, [3734](#)
  - emplace\_hint, [3735](#)
  - empty, [3736](#)
  - end, [3736](#), [3737](#)
  - equal\_range, [3737](#), [3738](#)
  - erase, [3738](#)–[3740](#)

- find, [3740](#), [3742](#)
- get\_allocator, [3742](#)
- hash\_function, [3742](#)
- hasher, [3724](#)
- insert, [3743–3747](#)
- iterator, [3725](#)
- key\_eq, [3747](#)
- key\_equal, [3725](#)
- key\_type, [3725](#)
- load\_factor, [3748](#)
- local\_iterator, [3725](#)
- mapped\_type, [3726](#)
- max\_bucket\_count, [3748](#)
- max\_load\_factor, [3748](#)
- max\_size, [3749](#)
- operator=, [3749](#)
- operator[], [3750](#)
- pointer, [3726](#)
- reference, [3726](#)
- rehash, [3751](#)
- reserve, [3751](#)
- size, [3752](#)
- size\_type, [3726](#)
- swap, [3752](#)
- unordered\_map, [3727–3729](#)
- value\_type, [3727](#)
- std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc  
loc >, [3759](#)
- allocator\_type, [3762](#)
- begin, [3769](#), [3770](#)
- bucket\_count, [3770](#)
- cbegin, [3770](#)
- cend, [3771](#)
- clear, [3772](#)
- const\_iterator, [3762](#)
- const\_local\_iterator, [3762](#)
- const\_pointer, [3763](#)
- const\_reference, [3763](#)
- count, [3772](#)
- difference\_type, [3763](#)
- emplace, [3772](#)
- emplace\_hint, [3773](#)
- empty, [3773](#)
- end, [3774](#), [3775](#)
- equal\_range, [3775](#), [3776](#)
- erase, [3776–3778](#)
- find, [3778](#), [3779](#)
- get\_allocator, [3779](#)
- hash\_function, [3779](#)
- hasher, [3763](#)
- insert, [3780–3784](#)
- iterator, [3764](#)
- key\_eq, [3784](#)
- key\_equal, [3764](#)
- key\_type, [3764](#)
- load\_factor, [3784](#)
- local\_iterator, [3764](#)
- mapped\_type, [3765](#)
- max\_bucket\_count, [3784](#)
- max\_load\_factor, [3785](#)
- max\_size, [3785](#)
- operator=, [3786](#)
- pointer, [3765](#)
- reference, [3765](#)
- rehash, [3787](#)
- reserve, [3787](#)
- size, [3787](#)
- size\_type, [3765](#)
- swap, [3788](#)
- unordered\_multimap, [3766–3768](#)
- value\_type, [3766](#)
- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc  
>, [3795](#)
- allocator\_type, [3798](#)
- begin, [3804](#), [3805](#)
- bucket\_count, [3806](#)
- cbegin, [3806](#)
- cend, [3807](#)
- clear, [3807](#)
- const\_iterator, [3798](#)
- const\_local\_iterator, [3798](#)
- const\_pointer, [3799](#)
- const\_reference, [3799](#)
- count, [3808](#)
- difference\_type, [3799](#)
- emplace, [3808](#)
- emplace\_hint, [3809](#)
- empty, [3809](#)
- end, [3809](#), [3810](#)
- equal\_range, [3811](#)
- erase, [3812](#), [3814](#)
- find, [3816](#)
- get\_allocator, [3817](#)
- hash\_function, [3817](#)
- hasher, [3799](#)
- insert, [3817–3820](#)
- iterator, [3800](#)
- key\_eq, [3820](#)
- key\_equal, [3800](#)
- key\_type, [3800](#)
- load\_factor, [3821](#)
- local\_iterator, [3800](#)
- max\_bucket\_count, [3821](#)
- max\_load\_factor, [3821](#)
- max\_size, [3822](#)
- operator=, [3822](#)
- pointer, [3801](#)
- reference, [3801](#)

- rehash, [3823](#)
- reserve, [3823](#)
- size, [3824](#)
- size\_type, [3801](#)
- swap, [3824](#)
- unordered\_multiset, [3802–3804](#)
- value\_type, [3801](#)
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3830](#)
  - allocator\_type, [3833](#)
  - begin, [3840](#), [3841](#)
  - bucket\_count, [3841](#)
  - cbegin, [3841](#)
  - cend, [3842](#)
  - clear, [3843](#)
  - const\_iterator, [3833](#)
  - const\_local\_iterator, [3834](#)
  - const\_pointer, [3834](#)
  - const\_reference, [3834](#)
  - count, [3843](#)
  - difference\_type, [3834](#)
  - emplace, [3843](#)
  - emplace\_hint, [3844](#)
  - empty, [3845](#)
  - end, [3845](#), [3846](#)
  - equal\_range, [3846](#), [3847](#)
  - erase, [3847–3849](#)
  - find, [3849](#), [3851](#)
  - get\_allocator, [3851](#)
  - hash\_function, [3851](#)
  - hasher, [3835](#)
  - insert, [3852–3855](#)
  - iterator, [3835](#)
  - key\_eq, [3855](#)
  - key\_equal, [3835](#)
  - key\_type, [3835](#)
  - load\_factor, [3855](#)
  - local\_iterator, [3836](#)
  - max\_bucket\_count, [3855](#)
  - max\_load\_factor, [3856](#)
  - max\_size, [3856](#)
  - operator=, [3857](#)
  - pointer, [3836](#)
  - reference, [3836](#)
  - rehash, [3858](#)
  - reserve, [3858](#)
  - size, [3858](#)
  - size\_type, [3836](#)
  - swap, [3859](#)
  - unordered\_set, [3837–3839](#)
  - value\_type, [3837](#)
- std::uses\_allocator< \_Tp, \_Alloc >, [3865](#)
- std::uses\_allocator< tuple< \_Types... >, \_Alloc >, [3866](#)
- std::valarray< \_Tp >, [3867](#)
  - valarray, [3869](#)
- std::vector< \_Tp, \_Alloc >, [3869](#)
  - \_M\_allocate\_and\_copy, [3877](#)
  - \_M\_range\_check, [3877](#)
  - ~vector, [3877](#)
  - assign, [3877](#), [3878](#)
  - at, [3879](#)
  - back, [3881](#)
  - begin, [3881](#), [3882](#)
  - capacity, [3882](#)
  - cbegin, [3882](#)
  - cend, [3882](#)
  - clear, [3883](#)
  - crbegin, [3883](#)
  - crend, [3883](#)
  - data, [3883](#)
  - emplace, [3884](#)
  - empty, [3884](#)
  - end, [3884](#), [3885](#)
  - erase, [3885](#), [3886](#)
  - front, [3886](#)
  - get\_allocator, [3887](#)
  - insert, [3887–3889](#)
  - max\_size, [3890](#)
  - operator=, [3890](#), [3891](#)
  - operator[], [3891](#), [3892](#)
  - pop\_back, [3892](#)
  - push\_back, [3892](#)
  - rbegin, [3893](#)
  - rend, [3893](#)
  - reserve, [3894](#)
  - resize, [3894](#), [3895](#)
  - shrink\_to\_fit, [3895](#)
  - size, [3895](#)
  - swap, [3896](#)
  - vector, [3873–3876](#)
- std::vector< bool, \_Alloc >, [3902](#)
- std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3906](#)
  - \_M\_buf\_locale, [3925](#)
  - \_M\_in\_beg, [3926](#)
  - \_M\_in\_cur, [3926](#)
  - \_M\_in\_end, [3926](#)
  - \_M\_out\_beg, [3926](#)
  - \_M\_out\_cur, [3926](#)
  - \_M\_out\_end, [3927](#)
  - \_\_streambuf\_type, [3908](#)
- char\_type, [3908](#)
- eback, [3910](#)
- egptr, [3911](#)
- epptr, [3911](#)
- gbump, [3911](#)
- getloc, [3912](#)
- gp\_ptr, [3912](#)
- imbue, [3912](#)

- in\_avail, 3913
- int\_type, 3909
- off\_type, 3909
- overflow, 3913
- pbackfail, 3914
- pbase, 3914
- pbump, 3915
- pos\_type, 3909
- pptr, 3915
- pubimbue, 3915
- pubseekoff, 3916
- pubseekpos, 3916
- pubsetbuf, 3917
- pubsync, 3917
- sbumpc, 3917
- seekoff, 3917
- seekpos, 3918
- setbuf, 3918
- setg, 3918
- setp, 3919
- sgetc, 3919
- sgetn, 3920
- showmanyc, 3920
- snextc, 3920
- sputbackc, 3921
- sputc, 3921
- sputn, 3922
- state, 3922
- sungetc, 3922
- sync, 3923
- traits\_type, 3909
- uflow, 3923
- underflow, 3924
- wbuffer\_convert, 3910
- xsggetn, 3924
- xsgputn, 3925
- std::weak\_ptr<\_Tp>, 3927
- std::weibull\_distribution<\_RealType>, 3928
  - a, 3930
  - b, 3930
  - max, 3930
  - min, 3930
  - operator(), 3931
  - operator==, 3932
  - param, 3931
  - reset, 3932
  - result\_type, 3929
- std::weibull\_distribution<\_RealType>::param\_type, 3243
- std::wstring\_convert<\_Codecvt, \_Elem, \_Wide\_alloc, \_Byte\_alloc>, 3932
  - converted, 3936
  - from\_bytes, 3936, 3937
  - state, 3937
  - to\_bytes, 3937, 3938
  - wstring\_convert, 3933, 3935
- std\_abs.h, 4278
- std\_function.h, 4278
- std\_mutex.h, 4279
- stdc++.h, 4280
- stddev
  - std::normal\_distribution<\_RealType>, 3105
- stdexcept, 4280
- stdio\_filebuf
  - \_\_gnu\_cxx::stdio\_filebuf<\_CharT, \_Traits>, 3499, 3500
- stdio\_filebuf.h, 4280
- stdio\_sync\_filebuf.h, 4281
- stdlib.h, 4281
- stdtr1c++.h, 4281
- stl\_algo.h, 4281
  - \_\_rotate, 4291, 4292
- stl\_algobase.h, 4292
- stl\_bvector.h, 4297
- stl\_construct.h, 4298
- stl\_deque.h, 4299
  - \_GLIBCXX\_DEQUE\_BUF\_SIZE, 4299
- stl\_function.h, 4300
- stl\_heap.h, 4302
- stl\_iterator.h, 4303, 4307
- stl\_iterator\_base\_funcs.h, 4308
- stl\_iterator\_base\_types.h, 4309
- stl\_list.h, 4310
- stl\_map.h, 4310
- stl\_multimap.h, 4311
- stl\_multiset.h, 4312
- stl\_numeric.h, 4313
- stl\_pair.h, 4314
- stl\_queue.h, 4315
- stl\_raw\_storage\_iter.h, 4315
- stl\_relops.h, 4316
- stl\_set.h, 4316
- stl\_stack.h, 4317
- stl\_tempbuf.h, 4318
- stl\_tree.h, 4318
- stl\_uninitialized.h, 4319
- stl\_vector.h, 4320
- stop\_token, 4321
- str
  - std::basic\_istream<\_CharT, \_Traits, \_Alloc>, 1727
  - std::basic\_ostringstream<\_CharT, \_Traits, \_Alloc>, 1860
  - std::basic\_stringbuf<\_CharT, \_Traits, \_Alloc>, 2026
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, 2084
  - std::match\_results<\_Bi\_iter, \_Alloc>, 2948
  - std::sub\_match<\_Biter>, 3562



- stream\_iterator.h, [4321](#)
- streambuf, [4321](#)
  - I/O, [128](#)
- streambuf.tcc, [4322](#)
- streambuf\_iterator.h, [4322](#)
- streambuf\_type
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2788](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3204](#)
- streamoff
  - std, [689](#)
- streampos
  - std, [689](#)
- streamsize
  - std, [689](#)
- stride
  - Numeric Arrays, [301](#)
- string, [4323](#), [4324](#), [4326](#)
  - Strings, [438](#)
- string\_conversions.h, [4326](#)
- string\_type
  - std::collate< \_CharT >, [2243](#)
  - std::collate\_byname< \_CharT >, [2250](#)
  - std::messages< \_CharT >, [2965](#)
  - std::money\_get< \_CharT, \_InIter >, [2979](#)
  - std::money\_put< \_CharT, \_OutIter >, [2985](#)
  - std::moneypunct< \_CharT, \_Intl >, [2992](#)
  - std::numpunct< \_CharT >, [3178](#)
- string\_view, [4327](#)
- string\_view.tcc, [4329](#)
- stringbuf
  - I/O, [128](#)
- stringfwd.h, [4330](#)
- Strings, [438](#)
  - string, [438](#)
  - u16string, [438](#)
  - u32string, [439](#)
  - wstring, [439](#)
- stringstream
  - I/O, [129](#)
- strstream, [4330](#)
- substr
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1043](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1974](#)
- subtract\_with\_carry\_engine
  - std::subtract\_with\_carry\_engine< \_UIntType, \_\_w, \_\_s, \_\_r >, [3568](#)
- subtractive\_rng
  - \_\_gnu\_cxx::subtractive\_rng, [3573](#)
- suffix
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2948](#)
- sum
  - Numeric Arrays, [301](#)
- sungetc
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2502](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3517](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3541](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1405](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1905](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2027](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3922](#)
- swap
  - \_\_gnu\_cxx, [512](#)
  - \_\_gnu\_cxx::\_\_versa\_string< \_CharT, \_Traits, \_Alloc, \_Base >, [1044](#)
  - \_\_gnu\_debug::basic\_string< \_CharT, \_Traits, \_Allocator >, [2005](#)
  - \_\_gnu\_parallel::iterator\_pair< \_Iterator1, \_Iterator2, \_IteratorCategory >, [1120](#), [1122](#)
  - \_\_gnu\_pbds::sample\_probe\_fn, [3384](#)
  - \_\_gnu\_pbds::sample\_range\_hashing, [3386](#)
  - \_\_gnu\_pbds::sample\_ranged\_hash\_fn, [3387](#)
  - \_\_gnu\_pbds::sample\_resize\_policy, [3392](#)
  - \_\_gnu\_pbds::sample\_resize\_trigger, [3397](#)
  - \_\_gnu\_pbds::sample\_size\_policy, [3398](#)
  - \_\_gnu\_pbds::sample\_update\_policy, [3403](#)
  - Futures, [105](#)
  - Numeric Arrays, [301](#)
  - Pointer Abstractions, [322](#)
  - Regular Expressions, [389](#), [390](#)
  - std, [787–791](#)
  - std::\_\_debug, [801](#)
  - std::basic\_regex< \_Ch\_type, \_Rx\_traits >, [1886](#)
  - std::basic\_string< \_CharT, \_Traits, \_Alloc >, [1974](#)
  - std::deque< \_Tp, \_Alloc >, [2435](#)
  - std::experimental::fundamentals\_v1::any, [1325](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2575](#)
  - std::function< \_Res(\_ArgTypes...) >, [2592](#)
  - std::list< \_Tp, \_Alloc >, [2858](#)
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2931](#)
  - std::match\_results< \_Bi\_iter, \_Alloc >, [2949](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3049](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3077](#)
  - std::pair< \_T1, \_T2 >, [3230](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3443](#)
  - std::shared\_ptr< \_Tp >, [3474](#)
  - std::sub\_match< \_Biter >, [3563](#), [3565](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2480](#)
  - std::unique\_lock< \_Mutex >, [3704](#)
  - std::unique\_ptr< \_Tp, \_Dp >, [3712](#)
  - std::unique\_ptr< \_Tp[], \_Dp >, [3719](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3752](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3788](#)

- std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3824
- std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3859
- std::vector< \_Tp, \_Alloc >, 3896
- Type-safe container of any type, 463
- Utilities, 481, 482
- swap\_ranges
  - Mutating, 217
- sync
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, 2502
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, 3517
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, 3542
  - std::basic\_filebuf< \_CharT, \_Traits >, 1405
  - std::basic\_fstream< \_CharT, \_Traits >, 1466
  - std::basic\_ifstream< \_CharT, \_Traits >, 1523
  - std::basic\_iostream< \_CharT, \_Traits >, 1620
  - std::basic\_istream< \_CharT, \_Traits >, 1674
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1727
  - std::basic\_streambuf< \_CharT, \_Traits >, 1906
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, 2027
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2084
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, 3923
- sync\_with\_stdio
  - std::basic\_fstream< \_CharT, \_Traits >, 1466
  - std::basic\_ifstream< \_CharT, \_Traits >, 1523
  - std::basic\_ios< \_CharT, \_Traits >, 1558
  - std::basic\_iostream< \_CharT, \_Traits >, 1620
  - std::basic\_istream< \_CharT, \_Traits >, 1674
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1728
  - std::basic\_ofstream< \_CharT, \_Traits >, 1773
  - std::basic\_ostream< \_CharT, \_Traits >, 1816
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1861
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2085
  - std::ios\_base, 2715
- syntax\_option\_type
  - std::regex\_constants, 841
- synth\_access\_traits
  - \_\_gnu\_pbds::detail::trie\_traits< Key, Mapped, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 3669
  - \_\_gnu\_pbds::detail::trie\_traits< Key, null\_type, \_ATraits, Node\_Update, pat\_trie\_tag, \_Alloc >, 3671
- synth\_access\_traits.hpp, 4330
- system\_category
  - Diagnostics, 81
- system\_error, 4331, 4332
- t
  - std::binomial\_distribution< \_IntType >, 2141
- table
  - std::ctype< char >, 2331
  - std::ctype\_byname< char >, 2384
- table\_size
  - std::ctype< char >, 2335
  - std::ctype\_byname< char >, 2388
- tag\_and\_trait.hpp, 4332
- Tags, 448
  - trivial\_iterator\_difference\_type, 448
- tags.h, 4334
- tan
  - Complex Numbers, 68
- tanh
  - Complex Numbers, 69
- target
  - std::function< \_Res(\_ArgTypes...) >, 2593
- target\_type
  - std::function< \_Res(\_ArgTypes...) >, 2593
- Technical Specifications, 449
- tellg
  - std::basic\_fstream< \_CharT, \_Traits >, 1467
  - std::basic\_ifstream< \_CharT, \_Traits >, 1524
  - std::basic\_iostream< \_CharT, \_Traits >, 1621
  - std::basic\_istream< \_CharT, \_Traits >, 1675
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1728
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2085
- tellp
  - std::basic\_fstream< \_CharT, \_Traits >, 1467
  - std::basic\_iostream< \_CharT, \_Traits >, 1621
  - std::basic\_ofstream< \_CharT, \_Traits >, 1773
  - std::basic\_ostream< \_CharT, \_Traits >, 1817
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1861
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2086
- temporary\_buffer
  - \_\_gnu\_cxx::temporary\_buffer< \_ForwardIterator, \_Tp >, 3578
- terminate
  - Exceptions, 91
- terminate\_handler
  - Exceptions, 88
- test
  - std::bitset< \_Nb >, 2164
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, 2480
- tgmath.h, 4334
- thin\_heap.hpp, 4334
- thousands\_sep
  - std::moneypunct< \_CharT, \_Intl >, 3002
  - std::moneypunct\_byname< \_CharT, \_Intl >, 3013

- std::numprint< \_CharT >, [3183](#)
- std::numprint\_byname< \_CharT >, [3189](#)
- thread, [4335](#)
- Threads, [450](#)
- throw\_allocator.h, [4336](#)
- throw\_with\_nested
  - Exceptions, [91](#)
- tie
  - std::basic\_fstream< \_CharT, \_Traits >, [1467](#), [1468](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1524](#), [1525](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1558](#), [1559](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1621](#), [1622](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1675](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1729](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1774](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1817](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1861](#), [1862](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2086](#)
  - Utilities, [483](#)
- Time, [451](#)
  - duration\_cast, [454](#)
  - high\_resolution\_clock, [453](#)
  - hours, [453](#)
  - microseconds, [453](#)
  - milliseconds, [453](#)
  - minutes, [453](#)
  - nanoseconds, [454](#)
  - operator\*, [454](#), [455](#)
  - operator+, [455](#)
  - operator-, [456](#)
  - seconds, [454](#)
  - time\_point\_cast, [456](#)
- time
  - std::locale, [2873](#)
- time\_get
  - std::time\_get< \_CharT, \_InIter >, [3596](#)
- time\_members.h, [4337](#)
- time\_point\_cast
  - Time, [456](#)
- time\_put
  - std::time\_put< \_CharT, \_OutIter >, [3625](#)
- tinyness\_before
  - std::\_\_numeric\_limits\_base, [954](#)
  - std::numeric\_limits< \_Tp >, [3157](#)
- TLB\_size
  - \_\_gnu\_parallel::Settings, [1287](#)
- to\_array
  - Array creation functions, [10](#)
- to\_bytes
  - std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_al-loc, \_Byte\_alloc >, [3937](#), [3938](#)
- to\_string
  - std::bitset< \_Nb >, [2165](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2481](#)
- to\_ullong
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2481](#)
- to\_ulong
  - std::bitset< \_Nb >, [2165](#)
  - std::tr2::dynamic\_bitset< \_WordT, \_Alloc >, [2481](#)
- tolower
  - std, [792](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [912](#)
  - std::ctype< \_CharT >, [2314](#), [2315](#)
  - std::ctype< char >, [2331](#), [2332](#)
  - std::ctype< wchar\_t >, [2350](#), [2351](#)
  - std::ctype\_byname< \_CharT >, [2368](#)
  - std::ctype\_byname< char >, [2384](#), [2385](#)
- top
  - std::priority\_queue< \_Tp, \_Sequence, \_Compare >, [3288](#)
  - std::stack< \_Tp, \_Sequence >, [3496](#)
- toupper
  - std, [792](#)
  - std::\_\_ctype\_abstract\_base< \_CharT >, [914](#)
  - std::ctype< \_CharT >, [2315](#), [2316](#)
  - std::ctype< char >, [2332](#), [2333](#)
  - std::ctype< wchar\_t >, [2351](#), [2352](#)
  - std::ctype\_byname< \_CharT >, [2370](#)
  - std::ctype\_byname< char >, [2385](#), [2386](#)
- TR1 Mathematical Special Functions, [440](#)
  - assoc\_laguerre, [442](#)
  - assoc\_legendre, [442](#)
  - beta, [442](#)
  - comp\_ellint\_1, [443](#)
  - comp\_ellint\_2, [443](#)
  - comp\_ellint\_3, [443](#)
  - conf\_hyperg, [443](#)
  - cyl\_bessel\_i, [444](#)
  - cyl\_bessel\_j, [444](#)
  - cyl\_bessel\_k, [444](#)
  - cyl\_neumann, [444](#)
  - ellint\_1, [445](#)
  - ellint\_2, [445](#)
  - ellint\_3, [445](#)
  - expint, [445](#)
  - hermite, [446](#)
  - hyperg, [446](#)
  - laguerre, [446](#)
  - legendre, [446](#)
  - riemann\_zeta, [447](#)
  - sph\_bessel, [447](#)
  - sph\_legendre, [447](#)
  - sph\_neumann, [447](#)
- trace\_fn\_imps.hpp, [4338](#), [4339](#)
- Traits, [457](#)

- traits.hpp, [4339–4341](#)
- traits\_type
  - std::basic\_ios< \_CharT, \_Traits >, [1544](#)
  - std::basic\_istream< \_CharT, \_Traits >::sentry, [3412](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1892](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2789](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3199](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3204](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3909](#)
- transform
  - Mutating, [217, 218](#)
  - std::collate< \_CharT >, [2247](#)
  - std::collate\_byname< \_CharT >, [2253](#)
  - std::regex\_traits< \_Ch\_type >, [3359](#)
- transform\_minimal\_n
  - \_\_gnu\_parallel::Settings, [1287](#)
- transform\_primary
  - std::regex\_traits< \_Ch\_type >, [3360](#)
- translate
  - std::regex\_traits< \_Ch\_type >, [3360](#)
- translate\_nocase
  - std::regex\_traits< \_Ch\_type >, [3362](#)
- traps
  - std::\_\_numeric\_limits\_base, [954](#)
  - std::numeric\_limits< \_Tp >, [3157](#)
- tree
  - \_\_gnu\_pbds::tree< Key, Mapped, Cmp\_Fn, Tag, Node\_Update, \_Alloc >, [3636](#)
- tree\_policy.hpp, [4341](#)
- tree\_trace\_base.hpp, [4342](#)
- trie
  - \_\_gnu\_pbds::trie< Key, Mapped, ATraits, Tag, Node\_Update, \_Alloc >, [3652, 3653](#)
- trie\_policy.hpp, [4342](#)
- trie\_policy\_base.hpp, [4342](#)
- trie\_string\_access\_traits\_imp.hpp, [4343](#)
- trivial\_iterator\_difference\_type
  - Tags, [448](#)
- true\_type
  - Metaprogramming, [197](#)
- trunc
  - std::num\_punct< \_CharT >, [3183](#)
  - std::num\_punct\_byname< \_CharT >, [3189](#)
- trunc
  - std::basic\_fstream< \_CharT, \_Traits >, [1479](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1535](#)
  - std::basic\_ios< \_CharT, \_Traits >, [1568](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1633](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1685](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1739](#)
  - std::basic\_ofstream< \_CharT, \_Traits >, [1784](#)
  - std::basic\_ostream< \_CharT, \_Traits >, [1827](#)
  - std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >, [1872](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2097](#)
  - std::ios\_base, [2723](#)
- try\_lock
  - Mutexes, [224](#)
- try\_to\_lock
  - Mutexes, [225](#)
- tuple, [4343, 4345](#)
- tuple\_cat
  - Utilities, [483](#)
- type
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, cc\_hash\_tag, Policy\_TI >, [2282](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, gp\_hash\_tag, Policy\_TI >, [2283](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, list\_update\_tag, Policy\_TI >, [2283](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, ov\_tree\_tag, Policy\_TI >, [2284](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, rb\_tree\_tag, Policy\_TI >, [2285](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, Mapped, \_Alloc, splay\_tree\_tag, Policy\_TI >, [2286](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, cc\_hash\_tag, Policy\_TI >, [2287](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, gp\_hash\_tag, Policy\_TI >, [2288](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, list\_update\_tag, Policy\_TI >, [2288](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, ov\_tree\_tag, Policy\_TI >, [2289](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, pat\_trie\_tag, Policy\_TI >, [2290](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< Key, null\_type, \_Alloc, splay\_tree\_tag, Policy\_TI >, [2291](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binary\_heap\_tag, null\_type >, [2278](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch< \_VTp, Cmp\_Fn, \_Alloc, binomial\_heap\_tag, null\_type >, [2279](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch<

- \_VTp, Cmp\_Fn, \_Alloc, pairing\_heap\_tag, null\_type >, [2280](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, rc\_binomial\_heap\_tag, null\_type >, [2280](#)
  - \_\_gnu\_pbds::detail::container\_base\_dispatch<\_VTp, Cmp\_Fn, \_Alloc, thin\_heap\_tag, null\_type >, [2281](#)
  - \_\_gnu\_pbds::detail::default\_comb\_hash\_fn, [2396](#)
  - \_\_gnu\_pbds::detail::default\_eq\_fn< Key >, [2399](#)
  - \_\_gnu\_pbds::detail::default\_hash\_fn< Key >, [2400](#)
  - \_\_gnu\_pbds::detail::default\_probe\_fn< Comb\_Probe\_Fn >, [2402](#)
  - \_\_gnu\_pbds::detail::default\_resize\_policy< Comb\_Hash\_Fn >, [2403](#)
  - \_\_gnu\_pbds::detail::default\_trie\_access\_traits<std::basic\_string< Char, Char\_Traits, std::allocator< Char > >, char > >, [2404](#)
  - \_\_gnu\_pbds::detail::default\_update\_policy, [2405](#)
  - \_\_gnu\_pbds::detail::entry\_cmp< \_VTp, Cmp\_Fn, \_Alloc, true >, [2512](#)
  - std::aligned\_union< \_Len, \_Types >, [1301](#)
  - std::experimental::fundamentals\_v1::any, [1325](#)
- Type-safe container of any type, [459](#)
  - any\_cast, [460–462](#)
  - swap, [463](#)
- type\_traits, [4346](#), [4351](#), [4352](#)
  - \_GLIBCXX\_HAS\_NESTED\_TYPE, [4351](#)
- type\_traits.h, [4355](#)
- type\_utils.hpp, [4356](#)
- typeid, [4356](#)
- typeinfo, [4357](#)
- typelist.h, [4357](#)
- types.h, [4359](#)
- types\_traits.hpp, [4359](#)
- u16streampos
  - std, [689](#)
- u16string
  - Strings, [438](#)
- u32streampos
  - std, [689](#)
- u32string
  - Strings, [439](#)
- uflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2502](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3517](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3542](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1405](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1906](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2027](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3923](#)
- uncaught\_exception
  - Exceptions, [91](#)
- uncaught\_exceptions
  - Exceptions, [91](#)
- undeclare\_no\_pointers
  - Pointer Safety and Garbage Collection, [324](#)
- undeclare\_reachable
  - Pointer Safety and Garbage Collection, [324](#)
- underflow
  - \_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2503](#)
  - \_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3518](#)
  - \_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >, [3542](#)
  - std::basic\_filebuf< \_CharT, \_Traits >, [1406](#)
  - std::basic\_streambuf< \_CharT, \_Traits >, [1906](#)
  - std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2028](#)
  - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3924](#)
  - underlying\_type\_t
    - Metaprogramming, [197](#)
- unexpected
  - Exceptions, [91](#)
- unexpected\_handler
  - Exceptions, [89](#)
- unget
  - std::basic\_fstream< \_CharT, \_Traits >, [1468](#)
  - std::basic\_ifstream< \_CharT, \_Traits >, [1525](#)
  - std::basic\_iostream< \_CharT, \_Traits >, [1622](#)
  - std::basic\_istream< \_CharT, \_Traits >, [1676](#)
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, [1730](#)
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, [2087](#)
- Uniform Distributions, [464](#)
  - operator!=, [464](#), [465](#)
  - operator<<, [465](#)
  - operator>>, [466](#)
- uniform\_int\_dist.h, [4360](#)
- uniform\_int\_distribution
  - std::uniform\_int\_distribution< \_IntType >, [3697](#)
- uniform\_real\_distribution
  - std::uniform\_real\_distribution< \_RealType >, [3700](#)
- uninitialized\_copy
  - Memory, [185](#)
- uninitialized\_copy\_n
  - Memory, [186](#)
  - SGI, [401](#)
- uninitialized\_fill
  - Memory, [186](#)
- uninitialized\_fill\_n
  - Memory, [187](#)
- unique
  - Mutating, [219](#)
  - std::forward\_list< \_Tp, \_Alloc >, [2575](#)
  - std::list< \_Tp, \_Alloc >, [2858](#)
  - std::shared\_ptr< \_Tp >, [3474](#)

- unique\_copy
  - Mutating, 220
- unique\_copy.h, 4360
- unique\_copy\_minimal\_n
  - \_\_gnu\_parallel::Settings, 1287
- unique\_lock.h, 4361
- unique\_ptr
  - std::unique\_ptr< \_Tp, \_Dp >, 3707, 3708
  - std::unique\_ptr< \_Tp[], \_Dp >, 3714–3716
- unique\_ptr.h, 4361
- unitbuf
  - std, 792
  - std::basic\_fstream< \_CharT, \_Traits >, 1479
  - std::basic\_ifstream< \_CharT, \_Traits >, 1535
  - std::basic\_ios< \_CharT, \_Traits >, 1568
  - std::basic\_iostream< \_CharT, \_Traits >, 1633
  - std::basic\_istream< \_CharT, \_Traits >, 1685
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1739
  - std::basic\_ofstream< \_CharT, \_Traits >, 1784
  - std::basic\_ostream< \_CharT, \_Traits >, 1828
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1872
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2097
  - std::ios\_base, 2724
- Unordered Associative, 468
- unordered\_map, 4362, 4363
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3727–3729
- unordered\_map.h, 4363
- unordered\_multimap
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, 3766–3768
- unordered\_multiset
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, 3802–3804
- unordered\_set, 4365, 4366
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, 3837–3839
- unordered\_set.h, 4366
- unsetf
  - std::basic\_fstream< \_CharT, \_Traits >, 1469
  - std::basic\_ifstream< \_CharT, \_Traits >, 1526
  - std::basic\_ios< \_CharT, \_Traits >, 1559
  - std::basic\_iostream< \_CharT, \_Traits >, 1623
  - std::basic\_istream< \_CharT, \_Traits >, 1676
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1730
  - std::basic\_ofstream< \_CharT, \_Traits >, 1774
  - std::basic\_ostream< \_CharT, \_Traits >, 1818
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1862
- std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2087
- std::ios\_base, 2715
- unshift
  - std::\_\_codecvt\_abstract\_base< \_InternT, \_ExternT, \_StateT >, 893
  - std::codecvt< \_InternT, \_ExternT, \_StateT >, 2208
  - std::codecvt< \_InternT, \_ExternT, encoding\_state >, 2214
  - std::codecvt< char, char, mbstate\_t >, 2219
  - std::codecvt< char16\_t, char, mbstate\_t >, 2224
  - std::codecvt< char32\_t, char, mbstate\_t >, 2229
  - std::codecvt< wchar\_t, char, mbstate\_t >, 2234
  - std::codecvt\_byname< \_InternT, \_ExternT, \_StateT >, 2240
- update\_fn\_imps.hpp, 4367
- upper\_bound
  - Binary Search, 44
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, 2932, 2933
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, 3049–3051
  - std::multiset< \_Key, \_Compare, \_Alloc >, 3078, 3079
  - std::set< \_Key, \_Compare, \_Alloc >, 3444, 3445
- uppercase
  - std, 792
  - std::basic\_fstream< \_CharT, \_Traits >, 1479
  - std::basic\_ifstream< \_CharT, \_Traits >, 1535
  - std::basic\_ios< \_CharT, \_Traits >, 1568
  - std::basic\_iostream< \_CharT, \_Traits >, 1633
  - std::basic\_istream< \_CharT, \_Traits >, 1686
  - std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >, 1740
  - std::basic\_ofstream< \_CharT, \_Traits >, 1785
  - std::basic\_ostream< \_CharT, \_Traits >, 1828
  - std::basic\_ostringstream< \_CharT, \_Traits, \_Alloc >, 1872
  - std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >, 2097
  - std::ios\_base, 2724
- use\_count
  - std::shared\_ptr< \_Tp >, 3474
- use\_facet
  - Locales, 145
  - std::locale, 2870
  - std::locale::id, 2684
- Utilities, 469
  - \_\_addressof, 474
  - \_\_invoke, 474
  - addressof, 474
  - forward, 475
  - forward\_as\_tuple, 475
  - get, 476, 477
  - make\_pair, 478



- move, [478](#)
- move\_if\_noexcept, [479](#)
- operator!=, [480](#)
- operator<, [480](#)
- operator<=, [480](#)
- operator>, [481](#)
- operator>=, [481](#)
- operator==, [480](#)
- pair, [472](#)
- piecewise\_construct, [483](#)
- swap, [481](#), [482](#)
- tie, [483](#)
- tuple\_cat, [483](#)
- utility, [4368](#), [4369](#)
- valarray, [4370](#)
  - Numeric Arrays, [267–269](#)
  - std::valarray< \_Tp >, [3869](#)
- valarray\_after.h, [4374](#)
- valarray\_array.h, [4384](#)
- valarray\_array.tcc, [4392](#)
- valarray\_before.h, [4393](#)
- valid\_prefix
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_citer< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, [1177](#)
  - \_\_gnu\_pbds::detail::pat\_trie\_base::\_Node\_iter< Node, Leaf, Head, Inode, \_Cliterator, Iterator, \_Alloc >, [1182](#)
- value
  - std::regex\_traits< \_Ch\_type >, [3362](#)
- value\_comp
  - std::map< \_Key, \_Tp, \_Compare, \_Alloc >, [2934](#)
  - std::multimap< \_Key, \_Tp, \_Compare, \_Alloc >, [3051](#)
  - std::multiset< \_Key, \_Compare, \_Alloc >, [3079](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3445](#)
- value\_compare
  - std::set< \_Key, \_Compare, \_Alloc >, [3422](#)
- value\_type
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_const\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [2107](#)
  - \_\_gnu\_pbds::detail::bin\_search\_tree\_node\_it< Node, Const\_Iterator, Iterator, \_Alloc >, [2113](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2125](#)
  - \_\_gnu\_pbds::detail::binary\_heap\_point\_const\_iterator< Value\_Type, Entry, Simple, \_Alloc >, [2130](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_const\_iterator< Node, \_Alloc >, [2802](#)
  - \_\_gnu\_pbds::detail::left\_child\_next\_sibling\_heap\_node\_point\_iterator< Node, \_Alloc >, [2808](#)
  - std::allocator\_traits< \_Alloc >, [1309](#)
  - std::allocator\_traits< allocator< \_Tp > >, [1316](#)
  - std::back\_insert\_iterator< \_Container >, [1370](#)
  - std::complex< \_Tp >, [2258](#)
  - std::front\_insert\_iterator< \_Container >, [2583](#)
  - std::insert\_iterator< \_Container >, [2699](#)
  - std::istream\_iterator< \_Tp, \_CharT, \_Traits, \_Dist >, [2784](#)
  - std::istreambuf\_iterator< \_CharT, \_Traits >, [2789](#)
  - std::iterator< \_Category, \_Tp, \_Distance, \_Pointer, \_Reference >, [2795](#)
  - std::ostream\_iterator< \_Tp, \_CharT, \_Traits >, [3199](#)
  - std::ostreambuf\_iterator< \_CharT, \_Traits >, [3204](#)
  - std::raw\_storage\_iterator< \_OutputIterator, \_Tp >, [3323](#)
  - std::reverse\_iterator< \_Iterator >, [3371](#)
  - std::set< \_Key, \_Compare, \_Alloc >, [3423](#)
  - std::unordered\_map< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3727](#)
  - std::unordered\_multimap< \_Key, \_Tp, \_Hash, \_Pred, \_Alloc >, [3766](#)
  - std::unordered\_multiset< \_Value, \_Hash, \_Pred, \_Alloc >, [3801](#)
  - std::unordered\_set< \_Value, \_Hash, \_Pred, \_Alloc >, [3837](#)
  - variant, [4393](#)
  - vector, [4393–4395](#)
    - std::\_\_debug::vector< \_Tp, \_Allocator >, [3899](#)
    - std::vector< \_Tp, \_Alloc >, [3873–3876](#)
  - vector.tcc, [4395](#)
  - void\_pointer
    - \_\_gnu\_cxx::\_\_alloc\_traits< \_Alloc, typename >, [871](#)
    - std::allocator\_traits< \_Alloc >, [1309](#)
    - std::allocator\_traits< allocator< \_Tp > >, [1316](#)
  - void\_t
    - Metaprogramming, [197](#)
  - vstring.h, [4396](#)
  - vstring.tcc, [4398](#)
  - vstring\_fwd.h, [4399](#)
  - vstring\_util.h, [4400](#)
  - wbuffer\_convert
    - std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3910](#)
  - wcerr
    - std, [795](#)
  - wcin
    - std, [795](#)
  - wclog
    - std, [795](#)
  - wcout
    - std, [795](#)
  - wcregex\_token\_iterator
    - Regular Expressions, [350](#)
  - wcsub\_match
    - Regular Expressions, [350](#)
  - wfilebuf

- I/O, [129](#)
- wfstream
  - I/O, [129](#)
- what
  - \_\_gnu\_pbds::container\_error, [2292](#)
  - \_\_gnu\_pbds::insert\_error, [2696](#)
  - \_\_gnu\_pbds::join\_error, [2798](#)
  - \_\_gnu\_pbds::resize\_error, [3367](#)
  - Exceptions, [91](#)
  - std::bad\_alloc, [1372](#)
  - std::bad\_cast, [1375](#)
  - std::bad\_exception, [1376](#)
  - std::bad\_function\_call, [1377](#)
  - std::bad\_typeid, [1379](#)
  - std::bad\_weak\_ptr, [1380](#)
  - std::domain\_error, [2459](#)
  - std::experimental::filesystem::v1::filesystem\_error, [2538](#)
  - std::experimental::fundamentals\_v1::bad\_any\_cast, [1374](#)
  - std::experimental::fundamentals\_v1::bad\_optional\_access, [1378](#)
  - std::future\_error, [2603](#)
  - std::invalid\_argument, [2704](#)
  - std::ios\_base::failure, [2537](#)
  - std::length\_error, [2811](#)
  - std::logic\_error, [2875](#)
  - std::out\_of\_range, [3207](#)
  - std::overflow\_error, [3217](#)
  - std::range\_error, [3306](#)
  - std::regex\_error, [3345](#)
  - std::runtime\_error, [3382](#)
  - std::system\_error, [3577](#)
  - std::underflow\_error, [3694](#)
- widen
  - std::\_\_ctype\_abstract\_base<\_CharT>, [915](#)
  - std::basic\_fstream<\_CharT, \_Traits>, [1469](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1526](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1560](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1623](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1677](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1731](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1775](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1818](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1863](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2088](#)
  - std::ctype<\_CharT>, [2316](#), [2318](#)
  - std::ctype<char>, [2334](#)
  - std::ctype<wchar\_t>, [2352](#), [2353](#)
  - std::ctype\_byname<\_CharT>, [2371](#)
  - std::ctype\_byname<char>, [2387](#)
- width
  - std::basic\_fstream<\_CharT, \_Traits>, [1470](#)
  - std::basic\_ifstream<\_CharT, \_Traits>, [1527](#)
  - std::basic\_ios<\_CharT, \_Traits>, [1560](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1624](#)
  - std::basic\_istream<\_CharT, \_Traits>, [1677](#)
  - std::basic\_istreamstream<\_CharT, \_Traits, \_Alloc>, [1731](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1775](#), [1776](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1819](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1863](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2088](#)
  - std::ios\_base, [2715](#), [2716](#)
- wfstream
  - I/O, [129](#)
- wios
  - I/O, [129](#)
- wiostream
  - I/O, [130](#)
- wistream
  - I/O, [130](#)
- wistreamstream
  - I/O, [130](#)
- wofstream
  - I/O, [130](#)
- workstealing.h, [4400](#)
- wostream
  - I/O, [130](#)
- wostringstream
  - I/O, [131](#)
- wregex
  - Regular Expressions, [350](#)
- write
  - std::basic\_fstream<\_CharT, \_Traits>, [1471](#)
  - std::basic\_iostream<\_CharT, \_Traits>, [1625](#)
  - std::basic\_ofstream<\_CharT, \_Traits>, [1776](#)
  - std::basic\_ostream<\_CharT, \_Traits>, [1819](#)
  - std::basic\_ostreamstream<\_CharT, \_Traits, \_Alloc>, [1864](#)
  - std::basic\_stringstream<\_CharT, \_Traits, \_Alloc>, [2089](#)
- ws
  - std, [792](#)
- wsregex\_token\_iterator
  - Regular Expressions, [350](#)
- wssub\_match
  - Regular Expressions, [351](#)
- wstreambuf
  - I/O, [131](#)
- wstreampos
  - std, [690](#)
- wstring



Strings, [439](#)

wstring\_convert  
std::wstring\_convert< \_Codecvt, \_Elem, \_Wide\_al-  
loc, \_Byte\_alloc >, [3933](#), [3935](#)

wstringbuf  
I/O, [131](#)

wstringstream  
I/O, [131](#)

xalloc  
std::basic\_fstream< \_CharT, \_Traits >, [1471](#)  
std::basic\_ifstream< \_CharT, \_Traits >, [1528](#)  
std::basic\_ios< \_CharT, \_Traits >, [1561](#)  
std::basic\_iostream< \_CharT, \_Traits >, [1625](#)  
std::basic\_istream< \_CharT, \_Traits >, [1678](#)  
std::basic\_istreamstream< \_CharT, \_Traits, \_Alloc >,  
[1732](#)  
std::basic\_ofstream< \_CharT, \_Traits >, [1777](#)  
std::basic\_ostream< \_CharT, \_Traits >, [1820](#)  
std::basic\_ostreamstream< \_CharT, \_Traits, \_Alloc >,  
[1864](#)  
std::basic\_stringstream< \_CharT, \_Traits, \_Alloc >,  
[2089](#)  
std::ios\_base, [2716](#)

xsgn  
\_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2503](#)  
\_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3518](#)  
\_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >,  
[3543](#)  
std::basic\_filebuf< \_CharT, \_Traits >, [1406](#)  
std::basic\_streambuf< \_CharT, \_Traits >, [1907](#)  
std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2028](#)  
std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3924](#)

xspn  
\_\_gnu\_cxx::enc\_filebuf< \_CharT >, [2504](#)  
\_\_gnu\_cxx::stdio\_filebuf< \_CharT, \_Traits >, [3519](#)  
\_\_gnu\_cxx::stdio\_sync\_filebuf< \_CharT, \_Traits >,  
[3544](#)  
std::basic\_filebuf< \_CharT, \_Traits >, [1407](#)  
std::basic\_streambuf< \_CharT, \_Traits >, [1908](#)  
std::basic\_stringbuf< \_CharT, \_Traits, \_Alloc >, [2029](#)  
std::wbuffer\_convert< \_Codecvt, \_Elem, \_Tr >, [3925](#)

yield  
std::this\_thread, [857](#)